FINAL REPORT

SAR Remote Sensing Algorithms for Automated Extraction of Sea Ice Ridges and Leads

University of Michigan Project
030098

Office of Naval Research Grant:
N00014-92-J-6005

Prof. John F. Vesecky, Principal Investigator

Atmospheric, Oceanic and Space Science Dept.
The University of Michigan

August, 1997

REPORT DOCUMENTATION PAGE

030098 Vesecky

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	January, 1997	Technical Report, 8/26/92 - 8/25/95

4. TITLE AND SUBTITLE

SAR Remote Sensing Algorithms for Automated Extraction of Sea Ice Ridges & Leads

5. FUNDING NUMBERS

N00014-92-J-6005

6. AUTHOR(S)

John F. Vesecky & Jason M. Daida

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Atmospheric, Oceanic, and Space Sciences Department
The University of Michigan
2455 Hayward St.
Ann Arbor, MI 48109-2143

8. PERFORMING ORGANIZATION REPORT NUMBER

030098-97-1

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unlimited Distribution

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

Sea ice is of fundamental importance in weather, climate and other geophysical processes. It is also an important factor for naval operations in the polar regions, in particular regarding transport of personnel and material in regions where sea ice is likely to be found and assessment and prediction of acoustic environments in polar regions. Because sea ice has a large geographic extent and short time scale for variability synthetic aperture radar (SAR) is a valuable technique in studying sea ice. Automated interpretation techniques are required because of the large number and high information content of the SAR images becoming available. Here we report research on automated-computer-based techniques for such interpretation. The principle approach under this grant is to use genetic algorithms to implement the segmentation. This final report on ONR grant N00014-92-J-6005 contains a summary of research under the grant together with an appendix of the several research papers that were produced under this grant. The papers themselves as well as demonstrations of the algorithms can be accessed through the ACERS website at the University of Michigan http://www.sprl.umich.edu/acers/. Many paper have an electronic appendix accessed through this website. The algorithms discovered through the genetic programming paradigm are shown to be efficient in extracting ridges features from SAR images, even those with low-contrast. We also apply the genetic paradigm to the problem of segmenting a SAR image into ridges, ice and leads. The SAR interpretation algorithms to which this research contributes can assure that SAR image information is available in a timely manner for use in ice science and naval applications.

14. SUBJECT TERMS

Synthetic Aperture Radar, Genetic Paradigm, Automated Image Interpretation, Sea Ice, Polar

15. NUMBER OF PAGES

87

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNC	UNC	UNC	Unlimited

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

ABSTRACT

Sea ice is of fundamental importance in weather, climate and other geophysical processes. It is also an important factor for naval operations in the polar regions, in particular regarding transport of personnel and material in regions where sea ice is likely to be found and assessment and prediction of acoustic environments in polar regions. Ridges (and keels) in sea ice are important because they provide a rougher surface and thus a higher drag coefficient for both atmospheric winds and ocean currents. This impacts sea ice motion. In addition ridges and keels can impact travel on the ice and operations under the ice since they constitute barriers to easy surface travel over the ice and a hazard to operations in the water column just under the surface. Because sea ice has a large geographic extent and short time scale for variability synthetic aperture radar (SAR) is a valuable technique in studying sea ice, particularly since images can be collected through clouds and at night. SAR information on sea ice is available from several satellites (ERS-1 & 2, JERS-1 and Radarsat). Automated interpretation techniques are required because of the large number and high information content of the SAR images becoming available. Here we report research on automated-computer-based techniques for such interpretation. The general problem that we face is to extract geophysical information from one or more SAR images. The work reported here concerns automated extraction of ridges and leads in sea ice. This amounts to segmentation of a sea ice surface into ice, ridges and leads. The principle approach under this grant is to use genetic algorithms to implement the segmentation.

This final report on ONR grant N00014-92-J-6005 contains a summary of research under the grant together with an appendix of the several research papers that were produced under this grant. The papers themselves as well as demonstrations of the algorithms can be accessed through the ACERS website at the University of Michigan http://www.sprl.umich.edu/acers/. Many paper have an electronic appendix accessed through this website that allows the reader to run some of the algorithms himself and thus become more familiar with their operation. The algorithms discovered through the genetic programming paradigm are shown to be efficient in extracting ridges features from SAR images, even those with low-contrast. We also apply the genetic paradigm to the problem of segmenting a SAR image into ridges, ice and leads. The SAR interpretation algorithms to which this research contributes can assure that SAR image information is available in a timely manner for use in ice science and naval applications.

In this report we include a bibliography of the publications that were supported by this research grant. In addition we have constructed an appendix of papers published. An important feature of our research is that we have constructed a world wide web site where users can not only access the text of the papers and high quality images, but also exercise the algorithms via the internet. This site called the ACERS (Adaptive Computation for Environmental and Remote Sensing Sciences) web site is

http://www.sprl.umich.edu/acers/papers.html

The sponsors of this work at NRL Stennis (Drs. Al Pressman and Florence Fetterer) and at ONR (Drs. Chuck Luther and Frank Herr) have contributed strongly to the success of this work both through financial support and through inspiration and useful suggestions.

TABLE of CONTENTS

I. Introduction and Motivation for Sea Ice Research

Sea ice is of fundamental importance in weather, climate and other geophysical processes. It is also an important factor for naval operations in the polar regions, in particular regarding transport of personnel and material in regions where sea ice is likely to be found and assessment and prediction of acoustic environments in polar regions. Because sea ice has a large geographic extent and short time scale for variability synthetic aperture radar (SAR) is a valuable technique in studying sea ice. A SAR image is a 'radar picture' of a scene on the Earth's surface. The picture elements or pixels of the scene are a map of the radar reflectivity of the pixel's area on the surface. SAR images are particularly useful in that they can be collected through clouds and at night. Such a capability is clearly important in polar regions. SAR information on sea ice will be available from several satellites in the 1990's (ERS-1, JERS-1, Almaz, Radarsat and possibly an EOS SAR). Automated interpretation techniques are required because of the large number and high information content of the SAR images becoming available. We have worked to develop automated-computer-based techniques for such interpretation and make use of advanced methods and concepts in image processing, computer vision and artificial intelligence.

The general problem that we face is to extract geophysical information from one or more SAR images. The work reported here concerns extraction of ridge features from sea images using genetic paradigm, automatic programming methods. The research reported here contains important results for the solution of this problem. The SAR interpretation algorithms to which this research contributes assure that SAR image information is available in a timely manner for use in ice science and naval applications. Ideas in the results reported here were fundamental to both the general scheme of the ice ridge feature extraction algorithms as well as the use of genetic programming methods in image processing for sea ice remote sensing.

II. Research Objectives

The objectives of research reported here follow from the circumstances discussed above. The objectives can be summarized as follows:

1. Apply the genetic programming paradigm to develop an image processing algorithm for extracting bright curvilinear features (ridges) from SAR images of sea ice.

2. Apply the genetic programming paradigm to the segmentation of SAR sea ice images into areas of differing surface roughness

3. Apply the genetic programming paradigm to the evaluation of image processing algorithms for sea ice remote sensing using SAR images

4. Apply the genetic programming paradigm to the general problem of image processing for sea ice image interpretation

III. Research Results

The research results stemming from the objectives above can be summarized under four topics:

A. Genetic programming for extracting sea ice ridges from SAR images
B. Evaluation of sea ice interpretation methods
C. Genetic programming for sea ice remote sensing in general.

The most important progress has been made in the first two items.

A. Genetic programming for extracting sea ice ridges from SAR images

Research work on extracting ridges from SAR images of sea ice began with previous work by Vesecky et al. (1988). Genetic programming is a method for generating algorithms to accomplish a given task in a manner similar to natural selection in biology. A wide variety of algorithms are generated and tested against a criteria for success. Only the best survives the natural selection process. The skill and art of genetic programming are in generating the algorithms that are tested and testing these algorithms in an efficient manner. An excellent introduction to the concept of genetic programming and current techniques is given by Koza (1992). We have applied the automatic programming process using the biological metaphors of genetic crossover and natural selection. In the process we specified a limited set of functions that includes algebraic and nonlinear image operators. As a fitness test for the natural selection process we compare the automatically generated algorithm results for a SAR image of sea ice with results of a manual analysis by an experienced operator. The results of the automatic programming are impressive, but are still not a good as a manual analysis by an experienced operator. The resulting algorithm and its results on a SAR sea ice image are shown in the paper 1 of the Appendix (Daida, et al., 1995).

This work was pursued further in Daida et al. (1996a). Here we used genetic programming (GP) to complement the normal hypothesis-test derivation of such algorithms. The most successful solution consists of a standard GP technique with a dynamic fitness function. The results for SAR images of sea ice are shown in the paper 2 of the Appendix. They are a distinct improvement over the results of Daida et al. (1995).

B. Evaluation of sea ice interpretation methods

Here we evaluate of the GP generated algorithms described in section A above. The evaluation is done using a SAR image of an Arctic ice camp. Members of the expedition on the ice surveyed the area for ridge and rubble features in multiyear ice and mapped them. We then used the algorithm of Daida (1996a) above with low resolution ERS-1 SAR images and extracted the ridge features. The results shown in Daida et al. (1996b, paper 3 in the Appendix) show that the GP algorithm performs well on low-resolution ERS-1 SAR images.

C. Genetic programming for sea ice remote sensing in general

Work under this topic includes genetic programming methods that apply both to the problem of extracting ridges in sea ice and to the application of GP techniques to remote

sensing algorithms in general. The general techniques developed are summarized briefly as follows:

1. Preprocessing: to speed up the use of GP techniques the problem presented to the GP automated programming is truncated as much as is practical, e.g. computing texture measures for an image before beginning the GP process

2. Test points: use of manually interpreted test points in an image to serve and GP programming benchmarks

3. Dynamic Fitness: increase the rigor of the fitness test as the programming process proceeds, i.e. use a relatively simple fitness test at the beginning to weed out the weak algorithms and then use a more rigorous and hence more complicated fitness test to select from among the stronger algorithms

4. Chunking: to reduce the computational overhead we divided an image into smaller subimages (chunks) so that computation speed could be increased because the computation time was more than proportional to the image size

5. Use of C-language: Although LISP is very well suited to GP methods, C-language can speed up processing on images

6. Scaffolding: Rather than let GP attempt to generate a code ab initio we let the used intervene from time to time to evaluate the natural selection process and change the fitness test points or prune off classes of algorithm that are not practical.

The papers by Daida et al. (1996c and 199d) describe these GP techniques and illustrate them with examples. These two papers are included in the Appendix as papers 4 and 5.

IV. Conclusions

The primary conclusions of this research can be summarized as follows:

1. Genetic programming techniques can automatically generate algorithms for sea ice image interpretation that are as good or better than algorithms generated by the normal hypothesis-test scheme.

2. Evaluation of GP generated algorithms for extraction of sea ice ridges from SAR images shows that these algorithms perform well on ERS-1 low-resolution SAR images.

3. We have developed and applied a number of techniques to increase the speed of GP methods, including preprocessing, chunking, scaffolding, etc. This makes the use of GP algorithms more practical since computation time is greatly reduced.

REFERENCES

Koza, J. R., Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, Cambridge MA (1992

Vesecky, J. F., R. Samadani, J. M. Daida, M. P. Smith and R. N. Bracewell, "Remote sensing of sea ice motion using floe edge and pressure ridge features in SAR images", 417-418, **IGARSS' 88 Conf. Proc.**, SEA SP-284, European Space Agency, Paris (Sept, 1988).

V. BIBLIOGRAPHY

Journal Papers and Book Chapters:

Daida, J.M., J.D. Hommes, T.F. Bersano-Begey, S.J. Ross, and J.F. Vesecky, "Algorithm Discovery Using the Genetic Programming Paradigm: Extracting Low-Contrast Curvilinear Features from SAR Images of Arctic Ice," to appear in Advances in Genetic Programming 2, P. Angeline and K. Kinnear, Jr. (ed.) Cambridge: The MIT Press, 1996a. pp. 417-442. See also the e-appendix* for this paper.

Conference and Symposia Papers:

Daida, J.M., J.D. Hommes, S.J. Ross, A.D. Marshall, & J.F. Vesecky, Extracting Curvilinear Features from SAR Images of Arctic Ice: Algorithm Discovery Using the Genetic Programming Paradigm, **Proceedings of the IEEE International Geoscience and Remote Sensing Symposium,** Firenze, Italy, T. Stein (ed.), Washington: IEEE Press (1995)

Daida, J.M., R.G. Onstott, T.F. Bersano-Begey, S.J. Ross, and J.F. Vesecky, Ice Roughness Classification and ERS SAR Imagery of Arctic Sea Ice: Evaluation of Feature-Extraction Algorithms by Genetic Programming, 1520-1522, **Proceedings of the 1996 International Geoscience and Remote Sensing Symposium**, Washington: IEEE Press (1996b). See also the e-appendix* for this paper.

Daida, J.M., T.F. Bersano-Begey, S.J. Ross, and J.F. Vesecky, "Evolving Feature-Extraction Algorithms: Adapting Genetic Programming for Image Analysis in Geoscience and Remote Sensing, 2077-2079, **Proceedings of the 1996 International Geoscience and Remote Sensing Symposium**, Washington: IEEE Press (1996c). See also the e-appendix* for this paper.

Daida, J.M., T.F. Bersano-Begey, S.J. Ross, and J.F. Vesecky, Computer-Assisted Design of Image Classification Algorithms: Dynamic and Static Fitness Evaluations in a Scaffolded Genetic Programming Environment, 279-284, **Genetic Programming 1996: Proceedings of the First Annual Conference**, J.R. Koza, D.E. Goldberg, D.B. Fogel, and R.L. Riolo (eds.). Cambridge: The MIT Press (1996d). See also the e-appendix* for this paper.

*Papers with and e-appendix can be accessed through the ACERS (Adaptive Computation Environmental and Remote Sensing Sciences) web site (http://www.sprl.umich.edu/acers/papers.html).

APPENDIX of PAPERS

PUBLISHED under ONR SPONSORSHIP

1995-1996

Paper 1

Daida, J.M., J.D. Hommes, S.J. Ross, A.D. Marshall, & J.F. Vesecky, Extracting Curvilinear Features from SAR Images of Arctic Ice: Algorithm Discovery Using the Genetic Programming Paradigm, **Proceedings of the IEEE International Geoscience and Remote Sensing Symposium**, Firenze, Italy, T. Stein (ed.), Washington: IEEE Press (1995)

Preprint, *Proceedings of the 1995 International Geoscience and Remote Sensing Symposium: Quantitative Remote Sensing for Science and Applications*, 10-14 July1995, Firenze, Italy.

Extracting Curvilinear Features from Synthetic Aperture Radar Images of Arctic Ice: Algorithm Discovery Using the Genetic Programming Paradigm

Jason M. Daida*, Jonathan D. Hommes*, Steven J. Ross*, and John F. Vesecky**

*The University of Michigan, Artificial Intelligence Laboratory & Space Physics Research Laboratory
2455 Hayward Avenue, Ann Arbor, Michigan 48109-2143
(313) 747-4581 FAX (313) 764-5137 EMAIL: daida@eecs.umich.edu
**The University of Michigan, Dept AOSS, 2455 Hayward Ave., Ann Arbor, MI 48109-2143

Abstract—This paper focuses on how a method for automated programming (i.e., genetic programming) applies in the computer-aided discovery of algorithms that enhance and extract features from remotely sensed images. Highlighted as a case study is the use of this method in the problem of extracting pressure ridge features from ERS-1 SAR imagery; a problem for which there has been no known satisfactory solution.

1. INTRODUCTION

Pressure ridges in arctic ice are a significant geophysical feature in sea-ice research. [1] Pressure ridges (and their corresponding keels, which are below-water features) help to transfer kinetic energy from meteorological systems and polar oceanic currents to the ice pack. In particular, pressure ridges and keels significantly increase sea-ice drag coefficients, which subsequently affect sea-ice movement and deformation.

To observe meso-scale features such as pressure ridges, researchers have used satellite SAR (synthetic aperture radar) imagery. In such imagery, pressure ridges often appear as filamentary, curvilinear features of variable width. With ERS-1, pressure ridges have radar backscatter signatures that differ only slightly from non-ridged multiyear-ice signatures. Pressure ridges subsequently appear mostly as low-contrast features in ERS-1 imagery. Extraction of such features by hand for quantitative analysis has proven extremely time consuming and tedious. Furthermore, current algorithms have also been shown to be ineffective with either low- or high-resolution ERS-1 data products. [2]

Current algorithms (e.g., [1], which are based on line detection and morphology) partially extract ridge features not as curvilinear segments, but as disjointed short segments and points. Current algorithms also extract many other non-ridge features—also as short segments and points—which result in a significant amount of clutter. We have hypothesized that texturally-filtered images have sufficient information for clutter removal and curvilinear feature enhancement. However, the number of possible texturally filtered images is large; and a proper combination of them is unknown. Consequently, we have chosen the genetic programming paradigm as a way to derive and systematically test possible algorithmic solutions.

2. OVERVIEW OF GENETIC PROGRAMMING

The genetic programming paradigm is a bottom-up, unsupervised programming method by Koza [3] that uses the biological metaphors of genetic crossover and natural selection for automatic programming. This fairly recent method loosely belongs to the class of domain-independent techniques in artificial intelligence (including neural nets and genetic algorithms) and is distinct from top-down automatic programming methods using planning techniques (e.g., [4]). Genetic programming represents one of the few approaches that exploit combinatorial search and that are capable of automatically deriving non-trivial code [5]. The paradigm has already been applied to solve a wide variety of problems in a number of domains, including an image analysis problem using feature vectors that were previously derived from image data [6]. However, as far as we know, there exists no published account of applying the paradigm directly to remotely sensed image data for solving geoscience problems.

3. PROBLEM-SPECIFIC IMPLEMENTATION

As in most domain-independent techniques, genetic programming requires the specification of a problem-specific portion of the genetic programming code. We have specified a limited set of terminals that includes subsets from an image and its associated texturally filtered counterparts (e.g., mean, variance). We have also specified a limited set of functions that includes algebraic and nonlinear image operators. The most difficult specification has been in deriving an appropriate fitness measure. Table 1 shows our current specification, which describes this situation: given one (8× 8 pixel) training set, derive a nonlinear ROI (region-of-interest) filter for pressure ridges in an entire image.

To evaluate the code, all of the benchmark matrices (8×8 subimages of intensity data and corresponding texturally- filtered subimages) were replaced with full-sized image data.

4. RESULTS & DISCUSSION

Table 2 shows the current algorithm (in LISP) that has been derived under the specifications described in Section 3. It has not been simplified, which has resulted in occasional sections of redundant code (e.g., subtracting a matrix from itself). The par-

ticular run that developed this individual took about an hour on a Sparc 2.

Figs. 1c and 1d show the results of this code on a 128×128 pixel image. (At the time of this writing, we were not ready to execute the code over a full-size image. We are contemplating porting the code from LISP to parallel C++) Dark areas are the ROI in which there exists a high probability of finding a ridge.

Upon close inspection, the results show that the algorithm has correctly identified the possible ROI. On one hand, the result is surprising since only one benchmark was used: conventional wisdom suggests that multiple benchmarks are needed to automatically generate robust code. Indeed, of the 64 pixels the genetic program had available to it, only 18 pixels (hand-selected) were used to derive an appropriate fitness measure.

Table 1. Problem-Specific Characterization

Terminal Set	Consists of a list of n selected points from an 8×8 benchmark, as well as corresponding points from four 8×8 filtered subimages (5×5 Laplacian of a 3×3 Mean, 5×5 Laplacian, 3×3 Mean, 5×5 Mean), and a random floating-point constant.
Function Set	Arithmetic operators defined to operate on matrices and constants in any combination (+, -, *, /), and the threshold operator If-Less-Than-or-Equal (IFLTE).
Fitness Cases	List of n manually selected points from benchmark. Pixels that are part of pressure ridge features are given a gray-scale value of 255 and all other non-feature pixels take a value of zero.
Raw Fitness	The sum over all n fitness cases of the difference between the target value and the output value for a given individual.
Standard Fitness	(Number of fitness cases, n) - (Number of hits).
Hits	The number of fitness cases for which an individual program's output is less than 50 away from the target gray-scale value of 255 or 0.
Success Predicate	The run ends when an individual program scores a hit on every fitness case or when the maximum number of generations is reached.

On the other hand, the result is typical of how genetic programming works. Genetic programming is parsimonious, especially when it comes to specifying functions and terminal sets. Either over-constraining a fitness function (e.g., by supplying too many data points to fit) or over-specifying the number and type of terminals and function sets has been found to be detrimental to algorithm discovery. In a sense, the paradigm forces a researcher to throw out as much of the extraneous information as is possible and to retain only a minimal set of functions and terminals from which to build a program. While this type of algorithmic behavior is unacceptable for many applications, it is acceptable, perhaps even desirable, in scaffolding the discovery process for other algorithms.

5. CONCLUSIONS & FUTURE WORK

This paper demonstrated how the genetic programming paradigm can apply to image processing. An open ended case example was studied, which involved deriving a nonlinear ROI filter for low-contrast curvilinear features. This paper has shown that the genetic programming paradigm can be used in the discovery process of deriving image processing algorithms.

Near-term future work features testing the robustness of the best-individual found so far. Other near-term work includes modifying the problem-specific code to include a dynamic fitness measure that uses generational benchmark swapping, which may help in deriving robust code.

We note that our specification for the problem-specific portion of the code is also generalizable to other similar problems that involve extracting features from image data. Finally, we believe that our specification is also generalizable for use with multispectral (and possibly multi-sourced) image data, since the multiple textural channels for each of our benchmarks can be replaced with a spectral channel.

ACKNOWLEDGMENTS

This research has been partially funded with grants from the Naval Research Laboratory (Stennis) and the Space Physics Research Laboratory (UM). We gratefully thank F. Fetterer, D. Rothrock, H. Stern, R. Shuchman, R. Onstott, and F. Tanis for our early discussions on pressure ridges. We are indebted to R. Riolo and J. Koza for conversations on GP.

BIBLIOGRAPHY

[1] Vesecky, J.F., M.P. Smith, R. Samadani, "Extraction of lead and ridge characteristics from SAR images of sea ice," *T-GRS*, 28:4, pp. 303-309, 1990.

[2] Gineris, D.J. & F.M. Fetterer, *The Joint Ice Center SAR Workstation. Algorithm Evaluation*, Memorandum Report 7019, Stennis Space Center, Naval Research Laboratory Detachment, March 1993.

[3] Koza, J.R., *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, Cambridge:MIT Press, 1992.

[4] Chien, S., "Using AI planning techniques to automatically generate image processing procedures: A preliminary report," *Proceedings of the Second International Conference on AI Planning Systems*, pp. 303-309, 1994.

[5] Holland, J., personal communications, 1994.

[6] Tackett, W.A., "Genetic programming for feature discovery and image discrimination," *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp. 219-224, 1993.

Table 2. Best-of-Run Individual

(÷ (+ *Laplacian-of-Mean* $Mean_{3\times3}$) (÷ (+ 1.9492742 $Mean_{3\times3}$) (× (− (× (÷ (≤ $Mean_{5\times5}$ -4.42039 (− (−$Mean_{3\times3}$ $Mean_{3\times3}$) (≤ (+ $Laplacian_{5\times5}$ $Mean_{3\times3}$) (≤ *Image* $Mean_{5\times5}$ *Image* $Mean_{5\times5}$) (÷$Laplacian_{5\times5}$ $Mean_{3\times3}$) (÷*Laplacian-of-Mean* $Mean_{5\times5}$))) $Mean_{3\times3}$) (÷ $Mean_{3\times3}$ -4.87104)) (÷ (÷ -4.721524 *Image*) (÷$Mean_{5\times5}$ 3.8854232))) (÷ (− (÷ (−$Mean_{3\times3}$ $Laplacian_{5\times5}$) (× $Laplacian_{5\times5}$ $Laplacian_{5\times5}$)) (≤ (+$Laplacian_{5\times5}$ $Mean_{3\times3}$) (− (× (× *Laplacian-of-Mean* $Mean_{3\times3}$) (− (÷ (−$Mean_{3\times3}$ $Laplacian_{5\times5}$) (× $Laplacian_{5\times5}$ $Laplacian_{5\times5}$)) (≤ (+$Laplacian_{5\times5}$ $Mean_{3\times3}$) (≤ *Image* $Mean_{5\times5}$ *Image* $Mean_{5\times5}$) (÷$Laplacian_{5\times5}$ $Mean_{3\times3}$) (÷ *Laplacian-of-Mean* $Mean_{5\times5}$)))) $Laplacian_{5\times5}$) (÷ $Laplacian_{5\times5}$ $Mean_{3\times3}$) (− (÷ (−$Mean_{3\times3}$ $Laplacian_{5\times5}$) (× $Laplacian_{5\times5}$ $Laplacian_{5\times5}$)) (≤ (+$Laplacian_{5\times5}$ $Mean_{3\times3}$) (≤ *Image* $Mean_{5\times5}$ *Image* $Mean_{5\times5}$) (÷$Laplacian_{5\times5}$ $Mean_{3\times3}$) (÷ *Laplacian-of-Mean* $Mean_{5\times5}$))))) (÷ (− (− (× (− (× (÷ (≤ $Mean_{5\times5}$ 4.42039 (≤ (+ 1.9492742 $Mean_{3\times3}$) (× (≤ *Image Image* $Laplacian_{5\times5}$ *Image*) (+ *Laplacian-of-Mean Laplacian-of-Mean*)) (+ *Image* $Mean_{3\times3}$) (× $Mean_{5\times5}$ $Mean_{5\times5}$)) $Mean_{3\times3}$) (÷ $Mean_{3\times3}$ -4.87104)) (÷ (÷ -4.721524 *Image*) (÷$Mean_{5\times5}$ 3.8854232))) (÷ (− (− (× -1.2301508 0.2565225) (≤ (+ 3.6199708 $Mean_{3\times3}$) (÷ *Image Image*) (÷ -2.134516 *Image*) (+ *Laplacian-of-Mean Image*))) (≤ (+$Laplacian_{5\times5}$ $Mean_{3\times3}$) (− $Laplacian_{5\times5}$ $Laplacian_{5\times5}$) (÷$Laplacian_{5\times5}$ $Mean_{3\times3}$) (− (÷ (−$Mean_{3\times3}$ $Laplacian_{5\times5}$) (+ *Laplacian-of-Mean* 3.8854232)) (≤ (+$Laplacian_{5\times5}$ $Mean_{3\times3}$) (≤ *Image* $Mean_{5\times5}$ *Image* $Mean_{5\times5}$) (÷$Laplacian_{5\times5}$ $Mean_{3\times3}$) (÷ *Laplacian-of-Mean* $Mean_{5\times5}$))))) (÷ (− (÷$Mean_{5\times5}$ 3.8854232) (÷ *Laplacian-of-Mean* $Mean_{3\times3}$)) (× (≤ 2.3869586 *Laplacian-of-Mean* (≤ $Mean_{3\times3}$ *Image* -3.3760195 $Laplacian_{5\times5}$) $Laplacian_{5\times5}$) (÷ -2.9275842 *Image*))))) (− $Laplacian_{5\times5}$ *Image*)) $Mean_{5\times5}$) (÷ *Laplacian-of-Mean* $Mean_{3\times3}$)) (× (≤ 2.3869586 *Laplacian-of-Mean* (≤ $Mean_{3\times3}$ *Image* -3.3760195 $Laplacian_{5\times5}$) $Laplacian_{5\times5}$) (÷ -2.9275842 *Image*))))) (−$Laplacian_{5\times5}$ *Image*))))

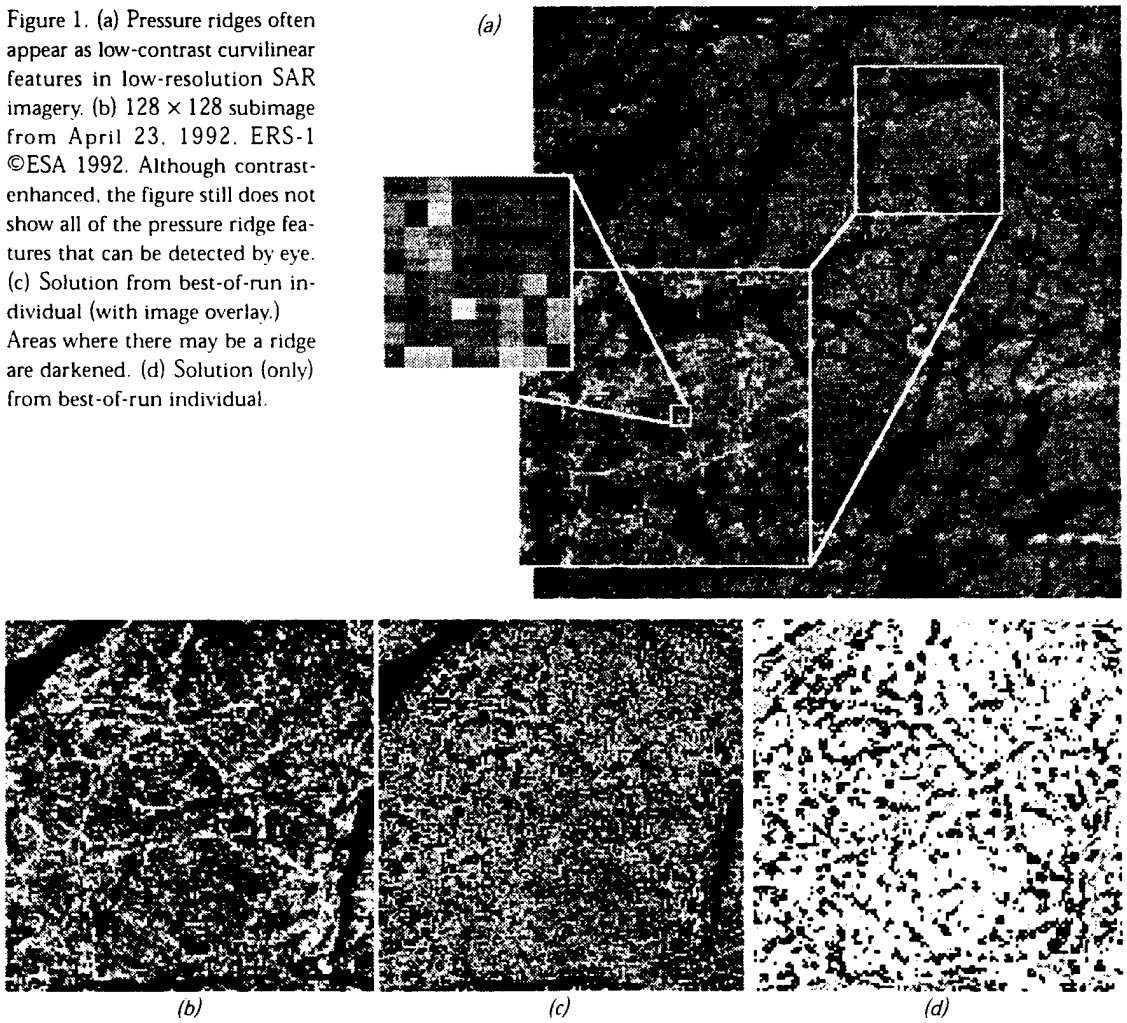


Figure 1. (a) Pressure ridges often appear as low-contrast curvilinear features in low-resolution SAR imagery. (b) 128 × 128 subimage from April 23, 1992, ERS-1 ©ESA 1992. Although contrast-enhanced, the figure still does not show all of the pressure ridge features that can be detected by eye. (c) Solution from best-of-run individual (with image overlay.) Areas where there may be a ridge are darkened. (d) Solution (only) from best-of-run individual.

Paper 2

Daida, J.M., J.D. Hommes, T.F. Bersano-Begey, S.J. Ross, and J.F. Vesecky, "Algorithm Discovery Using the Genetic Programming Paradigm: Extracting Low-Contrast Curvilinear Features from SAR Images of Arctic Ice," in Advances in Genetic Programming 2, P. Angeline and K. Kinnear, Jr. (ed.) Cambridge: The MIT Press, 1996a. pp. 417-442.

21 Algorithm Discovery Using the Genetic Programming Paradigm: Extracting Low-Contrast Curvilinear Features from SAR Images of Arctic Ice

Jason M. Daida, Jonathan D. Hommes, Tommaso F. Bersano-Begey, Steven J. Ross, and John F. Vesecky

This chapter discusses the application of genetic programming (GP) to image analysis problems in geoscience and remote sensing and describes how a GP can be adapted for processing large data sets (in our case, 1024 × 1024 pixel images plus texture channels). The featured problem is one that has not been adequately solved for this type of imagery. We describe the placement of GP in the overall scheme of algorithm discovery in geoscience image analysis and describe how GP complements a scientist's hypothesis-test derivation of such algorithms. The featured solution consists of a standard non-ADF GP that incorporates a dynamic fitness function.

21.1 Introduction

Computer-assisted design of image processing and analysis algorithms has helped scientists in five ways, all of which allow them to focus on their domain of expertise. First, repetitive processing sequences can be automated (i.e., task planning) [Chien 1994], [Matwin et al. 1995]. Second, fine-grained algorithmic detail—like data models and code-level programming—can be hidden to facilitate rapid prototyping of algorithms (i.e., visual programming) [Chang et al. 1990], [Konstatinides and Rasure 1994], [Rasure and Williams 1991]. Third, the extraction of patterns and features in an image can be automated. In some applications, patterns that may have escaped notice can be automatically sought for, extracted, and brought to a scientist's attention (e.g., data mining) [Hsu and Alexander 1994], [Koch and Moya 1994], [Nguyen and Huang 1994], [Openshaw 1995], [Tackett 1993]. Fourth, all algorithmic development can be hidden, which often involves automatic programming of image algebraic systems (e.g., automatic programming) [Barrera et al. 1994], [Chen 1992], [Vogt 1989]. Fifth, algorithmic development can be selectively exposed to guide a user, as well as to scaffold discovery of other algorithms. This chapter focuses on the fifth area and discusses the genetic programming paradigm [Koza 1992, 1994] as it applies to the computer-assisted design of classification filters for image processing. Highlighted as a case study is the use of genetic programming in the problem of extracting sea-ice pressure-ridge features from ERS SAR (synthetic aperture radar) imagery; a problem for which there has been no known satisfactory solution.

This chapter consists of seven sections. Section 21.1 covers background material, while Section 21.2 elaborates upon the use of GP as a method for scaffolding scientists in the algorithm-design process for image processing and analysis. Section 21.3 gives an overview of the implementation. Section 21.4 describes procedures and results concerning this chapter's case study. Sections 21.5 and 21.6 discuss the results, with the former emphasizing GP matters and the latter emphasizing domain-related issues. Section 21.7 concludes this chapter.

Preprint. To appear in *Advances in Genetic Programming II*, P. Angeline and K. Kinnear (ed.), Cambridge: The MIT Press, 1996

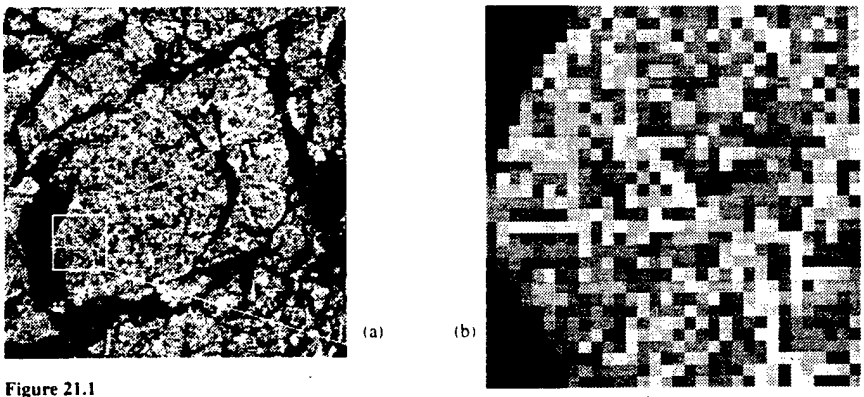


Figure 21.1
(a) Pressure ridges appear as low-contrast features in ERS SAR imagery. (a) 200 × 200 subimage from April 23, 1992. ERS-1 © ESA 1992. (b) Enlarged 32 × 32 subimage from (a). Although these figures are contrast-enhanced, they still do not show all of the pressure ridge features that can be detected by eye.

21.1.1 Domain of Case Study

The particular domain considered in this chapter involves extracting pressure-ridge features from ERS (European Remote Sensing Satellite) SAR images of arctic sea ice.[1] Pressure ridges are a significant geophysical feature in sea-ice research: ridges (and their corresponding keels, which are below-water features) help to transfer kinetic energy from meteorological systems and polar oceanic currents to the ice pack. In particular, pressure ridges and keels significantly increase sea-ice drag coefficients, which subsequently affect sea-ice movement and deformation [Burns and Wegener 1988], [Mellor 1986], [Vesecky et al. 1990].

Pressure ridges often result when thinner, first-year sea ice buckles under compression from thicker, multiyear ice. Since first-year ice often forms in long cracks (sometimes up to several hundreds of kilometers in length) between multiyear ice, pressure ridges usually appear as linear features on the ice cover. To a person standing on the ice, a new pressure ridge would appear as a several kilometer long hill about 5–10 m high and made of shattered, broken blocks about a meter thick [Mellor 1986]. When ridging occurs in a general area over an extended period of time, ridges consolidate to form rubble fields, which can also play a significant role in the transfer of kinetic energy.

When imaged with SAR, pressure ridges and rubble fields often appear to be brighter than their surrounding background (i.e., ridges and rubble have a slightly higher backscatter

[1] Like many other radars, SAR allows for all-weather, day or night operation. That is helpful in our case because sea ice occurs in areas of extended winter darkness and thick cloud cover.

signature). However, to say that pressure ridges appear as bright, curvilinear features in SAR imagery is to state something of an oversimplification. Pressure ridges appear as curvilinear features of varying width that often degenerate into blobs (rubble fields). Their radar backscatter, and hence their brightness in an image, greatly depend upon the particulars of a given SAR instrument.

When imaged with the SAR on board a European Space Agency ERS satellite (currently, ERS-1 and ERS-2), pressure ridges have radar backscatter signatures that differ only slightly from non-ridged multiyear-ice signatures. Pressure ridges subsequently appear mostly as low-contrast features in ERS imagery. (See Figure 21.1.) Extraction of such features by hand for quantitative analysis has proven extremely time consuming and tedious. Extraction of such features by current image processing algorithms has also proven untenable for several reasons, including: difficulties in specifying a shape grammar (as is possible in extracting roads from SAR imagery), low contrast (the sensor was optimized for other applications), clutter, and lack of ground truth (the phenomena occurs in remote locations). (*cf.* [Vesecky et al. 1990], which used data from a different SAR).

21.1.2 Objectives

We have hypothesized that there exists enough textural information in an ERS SAR image from which to design an ROI (region-of-interest) filter, a kind of spatial classifier that localizes the search area within which a particular feature can be found. However, the number of possible texture measures is large; and a proper combination of them is unknown.[2] Consequently, this chapter has several objectives:

- To describe the computer-assisted design of an image processing algorithm using a GP.
- To present a GP-derived solution for the case at hand.
- To offer an innovative approach in specifying fitness for problems using large data sets.

21.1.3 Contributions

This work makes the following contributions:

- Both this chapter and [Daida et al. 1995a] represent the first works featuring GP for image processing applications in geoscience and remote sensing. (Some of the special considerations of using image processing in these fields are given in Sections 21.2.2 and 21.3.)

[2] [Haralick and Shapiro 1992] gives an excellent overview in the treatment of texture in image processing. (In their work, they recognize thirteen major approaches to texture. Note that in just one of those techniques under those approaches—i.e., Haralick's gray-level co-occurrence matrices—consists of eight texture measures. The number of variations that can be tried with just these eight measures can significantly increase the total number of measures to consider in solving a given problem.) Throughout this chapter, we use the terms texture measure, texture filter, and texture channel as loosely interchangeable in deference to a textural energy approach.

• Presents an algorithm for extracting pressure-ridge features for ERS SAR imagery, a problem for which there has been no known satisfactory solution.

• Describes a general method for incorporating GP into scaffolded algorithm discovery for pattern recognition and classification of image data. We believe this method would also apply to image and texture types not presented in this chapter (e.g., multispectral images).

21.2 GP in Context of Scaffolding Algorithm Development

Algorithm design involves a wide range of activities that can span over several disciplines (e.g., computer science, mathematics, the domain science). Consequently, when we say that GP can be used for computer-assisted design of algorithms, we need to qualify where in the design process a method like GP would apply. This section describes our particular use of GP in the design process, notably in the scaffolding of algorithm discovery.

21.2.1 Scaffolding

Scaffolding—a term borrowed from education and learning theory—refers to supporting learners while they engage in activities that are normally beyond their reach [Brown and Palincsar 1989], [Vygotsky 1978], [Wood et al. 1975]. (The term metaphorically alludes to the temporary rigging that is erected around, say, a statue under construction.) Although scaffolding can take many forms, an accessible form is that of a mentor guiding an assistant. Current research in education includes extending the concept of scaffolding to include computer-assisted learning. (e.g., [Guzdial 1995], [Merrill and Reisner 1993])

In this chapter, we use the term *scaffolding* to include the computer-assisted support of experts (e.g., scientists, image analysts) who engage in activities that have an incompletely specified goal. (In a sense, even experts become learners when confronted with novel problems.) Incompletely specified goals are commonly encountered during the development of algorithms for interpreting remotely-sensed imagery, especially when:

• Firsthand knowledge (i.e., ground truth) of the area depicted in the scenes of interest is incomplete or missing.

• There is a high degree of ambiguity in how the phenomenon appears in a scene.

The case of extracting pressure-ridge features from ERS SAR imagery is typical of computational problems in remote sensing with incompletely specified goals for feature extraction [Cogalton 1991], [Daida et al. 1995b], [Lunetta et al. 1991]. Such computational problems often require extensive algorithm development, usually because a fair amount of domain-specific information needs to be built into the feature extraction algorithm. For that

reason, "off-the-shelf" software rarely succeeds and tailor-made code becomes the rule, rather than the exception.

With the overall task of discovering and creating a tailor-made algorithm comes a whole series of subtasks, of which only some of these subtasks can be replaced with GP. Consequently, before we can incorporate GP for computer-assisted support of algorithm discovery and development, we need to review some of these key subtasks.

21.2.2 GP and Scaffolding

For many applications, GP is used not as a scaffold for supervised design, but as a black box for unsupervised programming. On one hand, a black-box approach presupposes that performance metrics can be reasonably specified in advance of a GP run. There are three key subtasks that are subsequently involved: selection of components (i.e., function and terminal sets), selection of an evaluation metric (i.e., fitness function and fitness cases), and generation of an algorithm. With a black-box approach, a user supplies as input both algorithm components and performance metrics. GP, in turn, yields as output a program that fits the specifications that are implicit in the supplied metrics. In this way form follows function, as a GP "discovers" an appropriate form (an algorithm) to meet a prescribed functionality. Note that with this approach, functionality of a program is usually not in question—i.e., a programmer knows what she wants before the algorithm is ever written. Subsequent amendments to black-box inputs often end up serving as adjustments for getting GP to run on a particular problem.

On the other hand, a scaffolded approach does not necessarily presume that performance metrics are well-specified at the outset of a GP run. Under this approach, functionality of an algorithm can be very much in question—i.e., a programmer does not necessarily know what exactly needs to be accomplished. For example, specifying fitness cases for ridge extraction is anything but an absolute science. While it would be common for experts to agree that a ridge feature does exist in a particular location in an image, it would also be common for variations to exist on a pixel-by-pixel characterization of that ridge between those same experts. Furthermore, even with one expert examining a ridge, there is commonly a variation over time as that expert gains experience in seeing that type of feature.

An expert usually gains experience by formulating and then testing a hypothesis. For our domain problem, an expert would formulate a hypothesis that consists of classification rules (a kind of algorithm) categorizing ridge and non-ridge features in a sample subset. That hypothesis would be tested by applying those rules to an out-of-sample subset. Depending on that test's outcome, a hypothesis would be refined and another test would be run. Classification usually improves in this way. Discovery of what completely specifies a ridge feature becomes intrinsically linked with discovery of an algorithm that extracts such features.

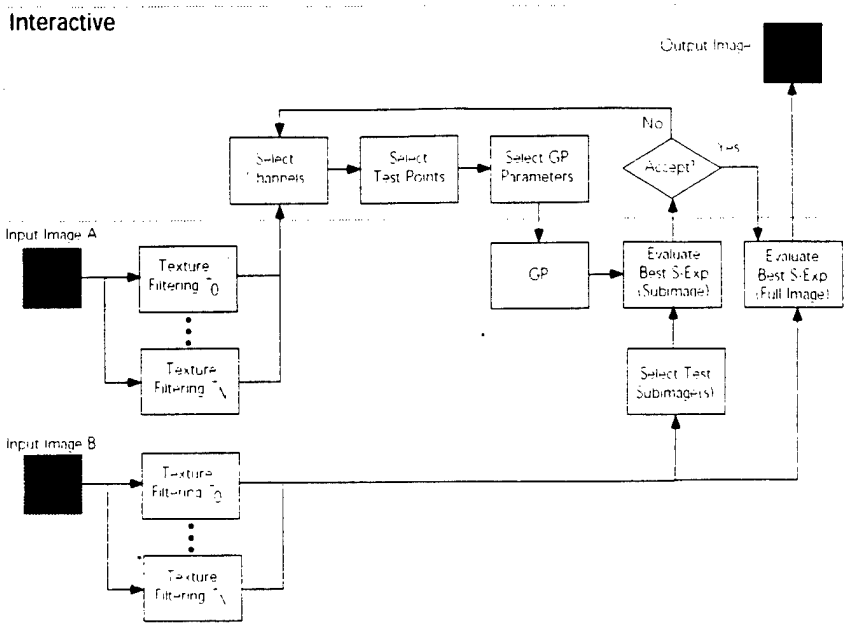


Figure 21.2
Flowchart of GP-scaffolded algorithm discovery for image analysis.

"Form" and "function" drive each other in a closed interactive cycle that, subsequently, involves several key subtasks: selection of components, selection of an evaluation metric (with fitness cases from a sample subset), generation of an algorithm, and an evaluation of that algorithm (with an out-of-sample subset).

We contend that GP can be used as a scaffold within this cycle of hypothesis-test by enabling the formulation of hypotheses of what would be involved in classification and also by facilitating rapid testing of these hypotheses to check for consequences. In comparison with a black-box approach, a user would supply as input both algorithm components and performance metrics. GP, in turn, would yield as output an algorithm that may fit the specifications that are implicit in the supplied metrics. By so doing, a hypothesis takes the form not as a full-blown algorithm, but as an algorithm sketch that consists of guesses and assumptions of what parts would make up such an algorithm. GP is then left to figure out the programming details. In contrast, a scaffolded approach requires a user to evaluate the performance of that algorithm on an out-of-sample subset to test not only the robustness of a GP-derived solution, but also the validity of the assumptions that went into a hypothesis. A user is then expected to modify a hypothesis should evidence so warrant. Unsupervised

programming and generation of a program then, becomes only one aspect in scaffolding a hypothesis-test cycle. The other relevant aspect includes educating an expert.

Figure 21.2 shows a flowchart of how we incorporated GP into this cycle. The upper half of that figure (boxed section) indicates those processes that are open for human intervention. The lower half indicates those processes that are intended mostly for autonomous operation. We designed this process to deliberately establish a problem-solving partnership between human and computer. The next section describes the components of Figure 21.2 in further detail.

21.3 Implementation Overview

Image analysis/processing problems in remote sensing and geosciences pose at least two major challenges to genetic programming: memory requirements and computational intensity.[1] Memory requirements are usually large: *one* image may typically have a size that measures in megabytes (e.g., a low-resolution data product) to tens of megabytes (e.g., a high-resolution satellite radar data product) to hundreds, even thousands of megabytes (e.g., a hyperspectral data product). Low resolution ERS SAR data products like the ones featured in this chapter are one megabyte in size (not including an image's meta-data). Computational intensities are reflected in processing times for typical geoscience image processing/analysis algorithms: many algorithms require CPU minutes; some, even CPU hours. Evaluation of even modestly sized populations of algorithms could require inordinate amounts of memory and processing times.

There exist at least two approaches to address these challenges: indirect processing using feature vectors and direct processing using training sets. Both approaches attempt to bring the memory and computation requirements down to GP-manageable levels: using feature vectors assumes that germane information can be described by an intermediate data product that is computationally derived and smaller in size than the original data set; using training sets assumes that there exists a small subset of the original data that adequately represents the image data *in toto*. The net effect of either approach is to constrain in size the amount of data used as input terminals for GP. We note that meeting this constraint is nontrivial and amounts to condensing information contained in several megabytes of image data into program inputs as small as a few hundred bytes.[2] While these constraints may seem severe, previous work has shown that working within these constraints is doable. [Tackett 1993] was the first to use genetic programming for image analysis problems. His work featured

[1] These challenges apply mostly to geoscience image processing. In comparison, many industrial (machine-vision) applications require video-frame rate processing times for images much smaller than a megabyte per frame.

[2] In some of our early work, we were working with compression ratios of roughly 60,000:1 [Daida et al. 1995a].

the indirect approach and applied GP to twenty statistical features that were derived from simulated infrared images. Likewise, [Nguyen and Huang 1994] also uses feature vectors to reduce memory and computational overhead.

This chapter describes an implementation based on the second approach. A key reason for using subimage training sets instead of feature vectors involves the presence or absence of a "grammar" that can be used to describe scene content. Man-made objects, like Nguyen and Huang's airplanes, can be decomposed into shape primitives, which can serve as a kind of grammar from which other airplanes can be modeled. Other man-made objects, like Tackett's infrared targets, can be decomposed into a series of metrics that can reasonably separate objects of interest from their background. In contrast, natural and geophysical objects often resist shape or metric decomposition because such decompositions usually suffer from a large number of exceptions. For that reason, "(machine) learning by (training set) example" is an option when working with natural objects. This chapter builds upon work previously presented in [Daida et al. 1995a]. The following sections present an overview of our implementation of GP fitness measures.

21.3.1 Terminals

In addition to a stochastic variable (floating point), all of the terminal sets in this chapter use the same combination of image data and texture filters: image data, $\text{Laplacian}_{5\times5}$ of $\text{Mean}_{3\times3}$, $\text{Laplacian}_{5\times5}$, $\text{Mean}_{3\times3}$, and $\text{Mean}_{5\times5}$.

This set of texture filters is based on Marr's work on representing an image [Marr 1982]. The first two filters ($\text{Laplacian}_{5\times5}$ of $\text{Mean}_{3\times3}$, which is an approximation to $\nabla^2 G$, and $\text{Laplacian}_{5\times5}$) are edge-detection type filters, which help to enhance ridges. The particular implementation of these filters involves rescaling the tonal values to fit an unsigned eight-bit gray scale. The last two filters ($\text{Mean}_{3\times3}$ and $\text{Mean}_{5\times5}$) are low-pass filters (for differing spatial scales), which help to enhance non-ridge features in first-year and multiyear ice. Note that subscripts appended to each filter name refer to the size of the convolution kernel associated with that filter.

21.3.2 Function Sets

Except where noted, we have used a similar function set that has been featured in many of the early test problems in GP [Koza 1992], including: arithmetic operators +, -, ÷, and ×, as well as the logical operator If-Less-Than-or-Equal-to (IFLTE, shown in this chapter as ≤). The selection of these functions were chosen in part because of their use in image processing of multichannel data (e.g., spectral ratioing, see [Lillesand and Kiefer 1987]). Unlike their mathematical counterparts, these operators have required a few algorithmic modifications to account for being closed under integer (unsigned eight-bit) arithmetic and mixed

Table 21.1
Arithmetic Operators.

Operation	Implementation	Arguments (Type Argument$_1$, Type Argument$_2$)		
		(Array, Array)	(Array, Number)	(Number, Number)
+ (Plus)	$(x + y)$ modulo 255	Adds arrays element by element	Adds number to each array element	Adds number to number
P (Plus)	$(x + y)$; if $(x + y) < 0$, 0; if $(x + y) > 255$, 255;	Adds arrays element by element	Adds number to each array element	Adds number to number
- (Minus)	cast $(x - y)$ as unsigned byte, then modulo 255	Subtract arrays element by element	Subtracts number from each array element	Subtracts number from number
M (Minus)	$(x - y)$; if $(x - y) < 0$, 0; if $(x - y) > 255$, 255;	Adds arrays element by element	Subtracts number to each array element	Adds number to number
× (Multi-plication)	(xy) modulo 255	Multiplies arrays element by element	Multiplies number to each array element	Multiplies number by number
/ (Protected Division)	(x/y) modulo 255; if $y=0$, $(x/y) = 1$	Divides arrays element by element	Divides number (array element) by each array element (number)	Divides number by number

data type usage (i.e., numbers and arrays). Note that we have used two versions of arithmetic + and –. Tables 21.1 and 21.2 describe the function set used by the problem-specific code described in this chapter.

21.3.3 Fitness

Fitness is computed over a training set that consists of image test points, as opposed to an image or even a subimage. For the purposes of this chapter, we define a test point as an array of pixel values at a point (x, y) in an image that has the form $c(x, y) \times i(x, y) \times T_0(x, y) \times \ldots \times T_N(x, y)$, where $c(.)$ is a manually derived classification of an image at location (x, y), $i(.)$ is the pixel intensity value for an image at location (x, y) and $T_N(.)$ is the pixel intensity value for the image that has been processed by the Nth filter in a bank of textural filters. Since in the case of extracting pressure ridges the relevant classifications are either "ridge" or "non-ridge" features, $c(.)$ is simply a Boolean quantity.

We have noted in [Daida et al. 1995a] that use of subimages for the case at hand, even with subimages as small as 8×8 pixels, produces two undesirable effects.

- Computational intensity dramatically increases, with the result of slowing execution duration down from CPU minutes (or hours) to CPU days.
- The accuracy of the resultant ROI algorithm significantly decreases.

[*] This is true for all cases of subtraction. However in retrospect, there should have been a distinction between Array-Number and Number-Array. The flaw went unnoticed until recently, in part because GP found a workaround.

Table 21.2
IFLTE operator. This operator has the form of (IFLTE Argument$_1$ Argument$_2$ Argument$_3$ Argument$_4$), i.e., IF Argument$_1$ ≤ Argument$_2$ THEN Argument$_3$ ELSE Argument$_4$.

Argument$_1$	Argument$_2$	Argument$_3$	Argument$_4$	Comments
Array	Array	Array	Array	Returns an array whose *n*th element is from either the *n*th element of Arg$_3$ or the *n*th element of Arg$_4$
Array	Array	Array	Number	Returns an array whose *n*th element is from either the *n*th element of Arg$_3$ or the number of Arg$_4$
Array	Array	Number	Array	Returns an array whose *n*th element is from either the number of Arg$_3$ or the *n*th element of Arg$_4$
Array	Array	Number	Number	Returns an array whose nth element is from either the number of Arg$_3$ or the number of Arg$_4$
Array	Number	Array	Array	Returns an array whose *n*th element is from either the *n*th element of Arg$_3$ or the *n*th element of Arg$_4$
Array	Number	Array	Number	Returns an array whose *n*th element is from either the *n*th element of Arg$_3$ or the number of Arg$_4$
Array	Number	Number	Array	Returns an array whose *n*th element is from either the number of Arg$_3$ or the *n*th element of Arg$_4$
Array	Number	Number	Number	Returns an array whose nth element is from either the number of Arg$_3$ or the number of Arg$_4$
Number	Array	Array	Array	Returns an array whose *n*th element is from either the *n*th element of Arg$_3$ or the *n*th element of Arg$_4$
Number	Array	Array	Number	Returns an array whose *n*th element is from either the *n*th element of Arg$_3$ or the number of Arg$_4$
Number	Array	Number	Array	Returns an array whose *n*th element is from either the number of Arg$_3$ or the *n*th element of Arg$_4$
Number	Array	Number	Number	Returns an array whose nth element is from either the number of Arg$_3$ or the number of Arg$_4$
Number	Number	Array	Array	Returns an array that is either Arg$_3$ or Arg$_4$
Number	Number	Array	Number	Returns either an array (Arg$_3$) or a number (Arg$_4$)
Number	Number	Number	Array	Returns either a number (Arg$_3$) or an array (Arg$_4$)
Number	Number	Number	Number	Returns a number that is either Arg$_3$ or Arg$_4$

A fitness input consisting of entire subimages apparently provides a GP with too much information at any one time, with much of the information being either extraneous or redundant. For that reason, we have used manually classified test points instead of subimage arrays. Note that a manual classification consists of a user's best guess of what classification to apply to each test point, which may not necessarily coincide with the underlying reality of each test point. (See also Section 21.5.)

21.3.4 Implementation Notes

Texturally filtered images that correspond to filters T_0 through T_N (where $N = 3$) were obtained using NIH Image* running on a Macintosh computer. Selection of test subimages and generation of concatenated image files that were used by the GP input- and output-wrap-

*NIH Image is a public domain image processing and analysis program by W. Rasband that is available via anonymous ftp at zippy.nihm.nih.gov in directory /pub/image.

pers were implemented largely with custom-made NIH Image macros.

All implementations of the code associated with the GP were done in LISP (Allegro Common LISP) and run on a Unix workstation (either a SunSPARC 2, SunSPARC 20 or HP 715) that typically had 32 MB RAM. These programs included custom-made input-terminal generators (which are implicit in the step "Select Test Points"), custom-made visualization-wrappers (which are implicit in the step "Evaluate Best S-Exp"), our fitness functions, and a slightly modified version of Koza's GP kernel (modifications were mostly for input/output). We note that the IFLTE function was done as a LISP macro, as opposed to a LISP function.

Even with 32 MB RAM, a workstation cannot process in LISP a full 1024 × 1024 eight-bit image without running into memory problems. For that reason, all low-resolution ERS data products were chunked into 128 × 128 pixel subimages in NIH Image (with a custom macro) before applying a GP-derived algorithm (an s-expression). The chunked subimages were reassembled in NIH Image (with another custom macro) after processing (i.e., during the step "Evaluate Best S-Exp").

One key feature in the GP-assisted design process lies in the placement of the feedback loop (see Figure 21.2). Note that for many GP applications, the acceptance or rejection of an algorithm can be largely determined by that algorithm's fitness measure. For our particular application, a fitness measure is only a rough indicator of performance, largely because the input terminals represent only a small fraction of an entire data set (i.e., many low-resolution data products). Consequently, an algorithm needs to be evaluated over a much larger data set than is represented by the training set of test points. For our purposes, we processed 128 × 128 pixel subimages to make intermediate decisions about algorithm performance, since that size subimage was about as a large a data set as our LISP visualization-wrapper could accept in one pass. In this way, feedback and the scaffolding of "what is important" in extracting ridge features was based on a qualitative assessment of output image data, rather than relying on a quantitative metric (like a fitness measure).

21.3.5 Image Data

The image data featured in this chapter is part of a larger series of temporal SAR data beginning in August 1991 and ending in July 1992. This data set describes the synoptic coverage of an area in the Beaufort Sea gyre (roughly 72° N, 140° W). The particular area and year of coverage were chosen to coincide with the LEADEX campaign. All the images shown in this chapter are low-resolution ERS-1 SAR data products, 1024 × 1024 pixels in size, eight-bit gray scale, calibrated, 100 m pixel size, and non-geocoded. Figure 21.3 shows the particular images that are featured in this chapter. Note that Figure 21.3b represents the

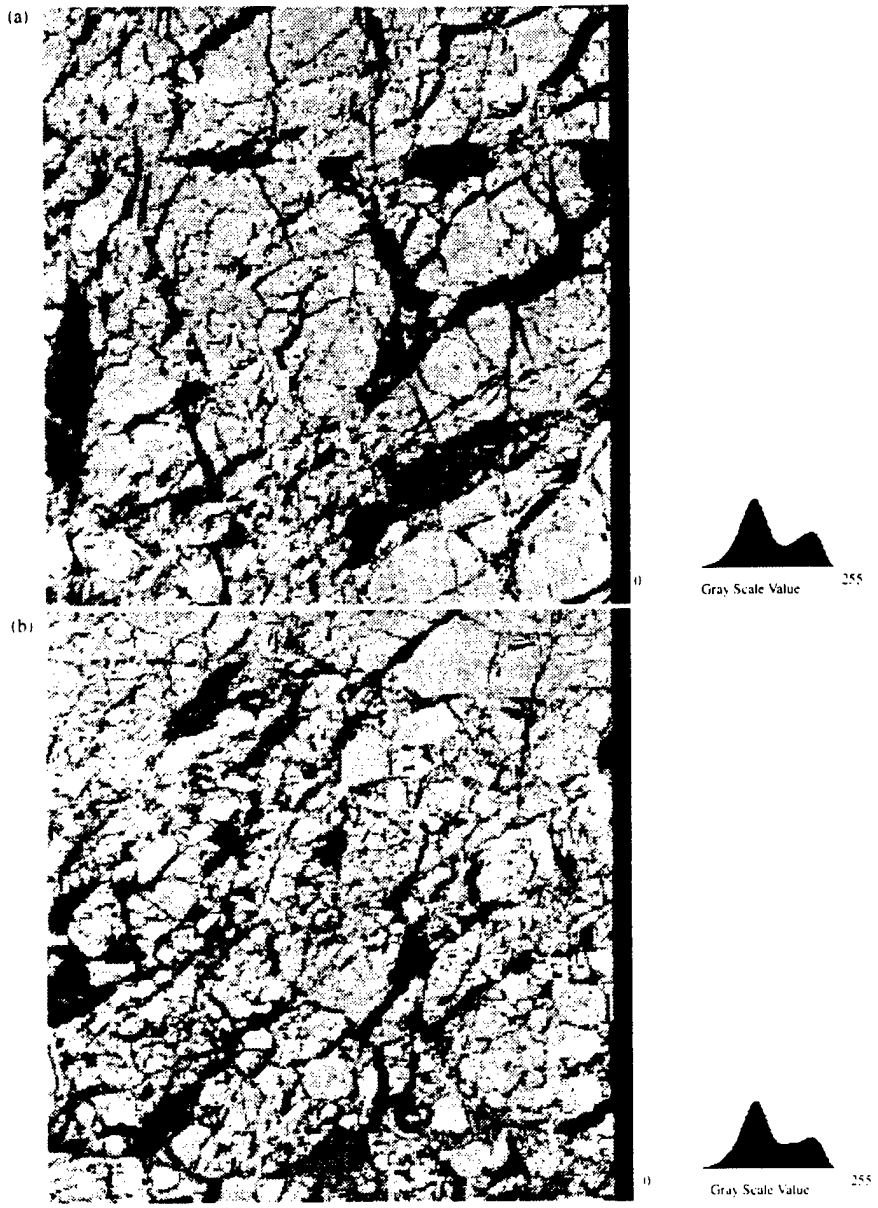


Figure 21.3
ERS-1 SAR images featured in this chapter. Both images are contrast-enhanced for publication and are 1024 × 1024 pixels in size, which is approximately equivalent to 100 × 100 sq km. The black stripe on the right of each image indicates an absence of image data for those corresponding locations. A gray-level histogram for each image is also shown. (a) March 2, 1992. ©ESA 1992. (b) April 23, 1992. ©ESA 1992.

image that contained fitness test points; Figure 21.3a represents the image that did not.

21.4 GP-Assisted Discovery Using A Dynamic Training Set

Our first work on using GP for this domain problem [Daida et al. 1995a] featured the use of a fixed training set, i.e., fitness evaluations were based upon a training set that did not change over the course of a GP run. Although the results were encouraging, a major shortfall in using a static training set had to do with the controllability and repeatability of evolving a desired result. A different choice of test points can and has yielded entirely different algorithms with widely varying degrees of success. For this and other reasons that are discussed in Section 21.5, we opted to go with a dynamic training set, i.e., fitness evaluations are based upon a training set that changes during the course of a GP run.

The particular strategy that we designed into the fitness function borrows from Goldberg's work in genetic algorithm classifiers [Goldberg 1989] and Holland's work in default hierarchies and induction [Holland et al. 1986]. The idea is fairly straightforward. A GP system starts a run with a training set that is relatively small and contains test points that should, in theory, be easy for the algorithm to score well. When an individual scores a certain number of hits, a few more points are added to the training set under evaluation. This process can continue until either an individual scores a maximum number of hits or a maximum number of generations has been reached. Not only has this strategy resulted in a better individual than described in [Daida et al. 1995a], but the overall process under this fitness function has proven to be more controllable than when using a static training set.

21.4.1 Procedure

Table 21.3 summarizes the problem-specific portion of the GP code described in this chapter. The other GP parameters that were used included: maximum number of generations equal to 30; size of population, 357; maximum depth of individuals 10; maximum depth of new subtrees for mutants, 4; maximum depth of individuals after crossover, 20; fitness-proportionate reproduction fraction, 0.1; crossover at any point fraction, 0.2, crossover at function points fraction, 0.7; selection method, fitness-proportionate; generation method, ramped-half-and-half. Note that the population size and number of generations used were modest and chosen in part so that we could complete a run on a SunSPARC 20 within a few CPU hours.

Figure 21.4 details our implementation of this fitness evaluation function by elaborating on the fitness set used to generate results shown later in this chapter. Figure 21.4a shows a portion of the image data plus associated texture images. The boxed areas show the loca-

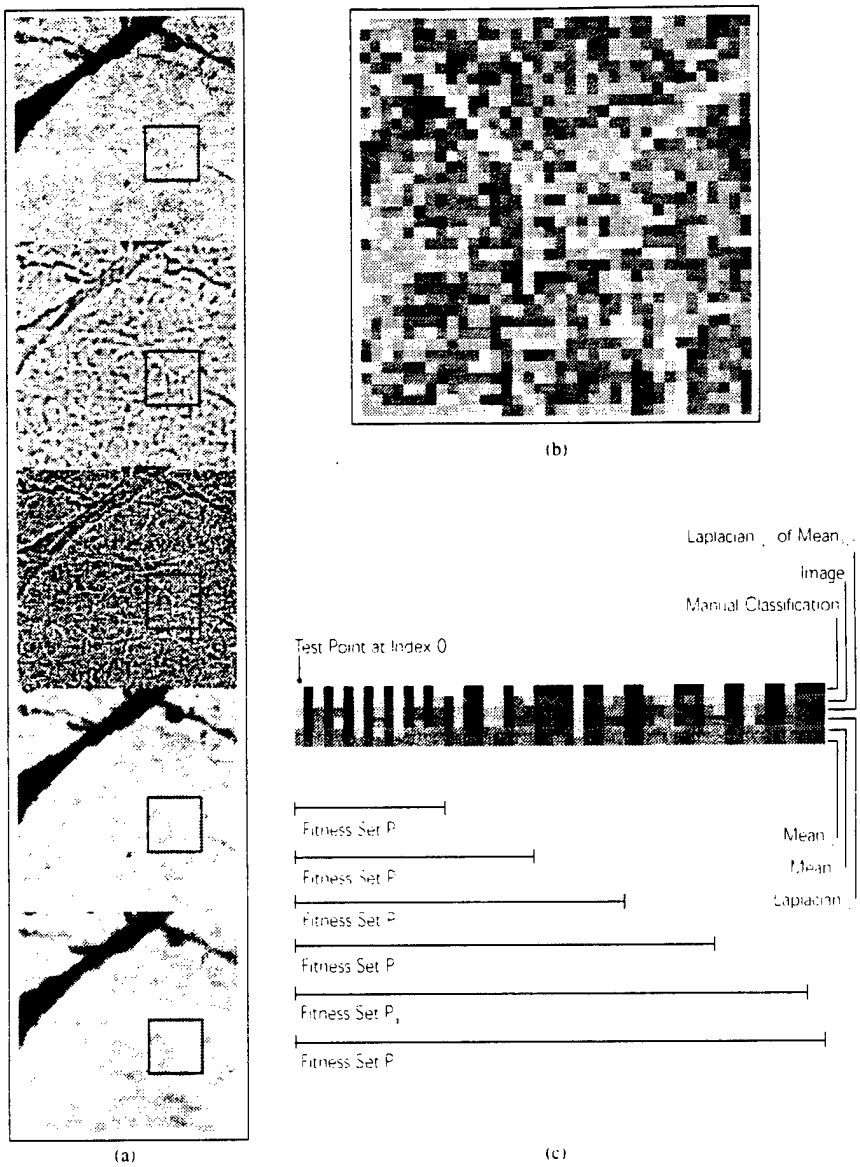


Figure 21.4
Fitness Set Construction. Starting from the upper left image: (a) 128 × 128 pixel ERS-1 SAR image (April 23, 1992) and associated layers of texturally filtered images (Laplacian$_{7\times7}$ of Mean$_{11\times11}$, Laplacian$_{7\times7}$, Mean$_{11\times11}$, and Mean$_{7\times7}$). (b) Overlay of test point locations. White points indicate ridge guesses, while black points indicate background guesses. (c) Detail of Fitness Set. Each row corresponds to either a manual classification, image data, or layer of texturally filtered image data. Each column corresponds to a test point. Test points are loaded into an evaluation function by sets, starting from P_1 to P_6. Loading of a set is triggered by criteria that are set by a user.

Table 21.3
Tableau for Case Study.

Terminal Set:	Consists of an array of size N of manually selected test points, which contains data from an image and its corresponding filtered versions (i.e., 5×5 Laplacian of a 3×3 Mean, 5×5 Laplacian, 3×3 Mean, 5×5 Mean); a random floating-point variable.
Function Set:	Arithmetic operators defined to operate on matrices and constants in any combination (+, -, ×, ÷, P, M), and the threshold operator If-Less-Than-or-Equal (≤).
Fitness Cases:	List of N manually selected control points from benchmark. Pixels that are part of pressure ridge features are given a gray-scale value of 255 and all other non-feature pixels take a value of zero. The number of current fitness cases increases dynamically (See Figure 21.4).
Raw Fitness:	The number of hits.
Standard Fitness:	(Number of current fitness cases, N) - (Number of hits).
Hits:	The number of fitness cases for which an individual program's output is less than 9 away from the target gray-scale value of 255 or 0.
Success Predicate:	The run ends when the maximum number of generations is reached.

tions of the 36 × 36 pixel area depicted in Figure 21.4b as it occurs in all five layers of either image data or texturally-filtered image data. Figure 21.4b shows the locations of the test points that were used and overlays those locations on the image-data layer. Also shown are the manual classifications associated with each point. Figure 21.4c shows the detail of the fitness set. There are 53 test points shown, which are arranged in six layers. Each layer corresponds to a channel of data (e.g., image data). The test points were manually ordered so that the easiest test points were loaded in for evaluation first. The bars showing the fitness set increments depict an initial starting set of fifteen test points, with four set increments of nine points apiece, and a fifth set increment of two points. In all there were at most six fitness sets total (Sets P_0 to P_5) that were considered during a single GP run.

We note that our particular implementation uses the following event to trigger the loading of the next fitness set: when a fitness evaluation for any one individual in a population is less than or equal to q standard fitness, load in the next increment of test points for the next generation to evaluate (e.g., to obtain the best-of-runs individual featured in Figure 21.7, we used a trigger value of six).

21.4.2 Results

Figure 21.5 shows the result of applying the best-of-runs individual to both the March 2 and April 23 images. In a qualitative examination of these results, we have found that the extracted features are well correlated with ridge and rubble features in both multiyear and first-year ice. Black (gray value 255) represents extracted pressure-ridge features. Varying shades of gray indicate the likelihood of a pressure ridge—the darker the gray, the higher the likelihood. The performance of this individual has been deemed better than the perfor-

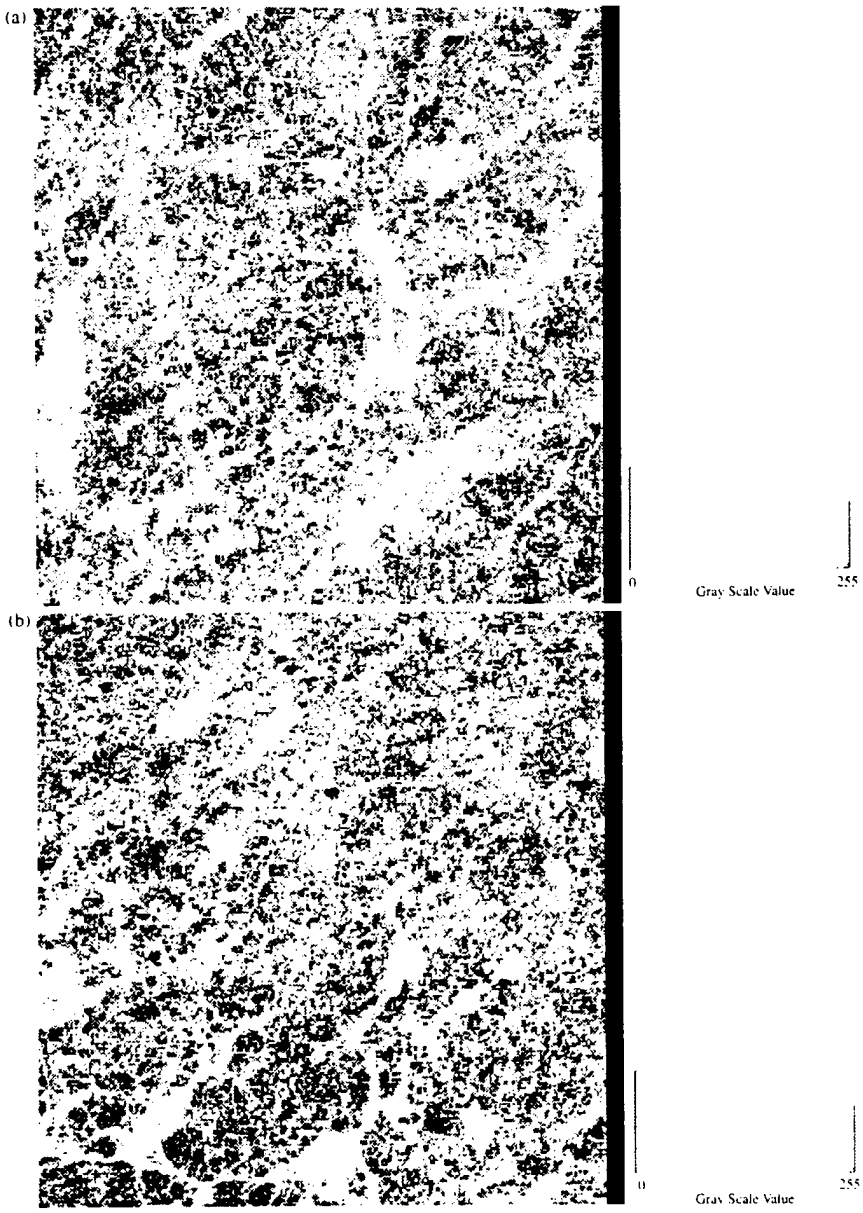


Figure 21.5
Images after processing with GP switch filter. Each image is 1024 × 1024 pixels. (100 × 100 sq km). A gray-level histogram for each image is also shown. The histograms show that the classifications are strongly bimodal: pixels with a value of 0 (white) correspond to non-ridge features, while pixels with values at or nearby 255 (black) correspond to ridge features. (a) March 2, 1992. (b) April 23, 1992.

mance of the individual described in [Daida et al. 1995a], since this algorithm (called a switch filter in this chapter) provided better connectivity of ridge features than the filter described in [Daida et al. 1995a] (called a stipple filter).

21.5 Discussion: GP & Scaffolding

Discussion of the results in Section 21.4 are divided into two sections. The first section, which is covered here, involves a discussion of the results from the standpoint of algorithm discovery and genetic programming. The second section (Section 21.6) involves a discussion of the results from the standpoint of the domain of polar oceanography.

21.5.1 Scaffolding and the Evolution of Hypotheses

The role of GP in scaffolding algorithm discovery has been to generate possible solutions based on a given hypothesis, which has usually taken the form of a strategy for selecting training sets. In some instances, these strategies affected the GP fitness evaluations function. In other instances, these strategies affected the kinds of test points that were eventually selected. The following paragraphs provide a synopsis of hypothesis evolution from our earlier work presented in [Daida et al. 1995a] to this present work. Described are only the major hypotheses—implicit are the numerous minor hypotheses subsumed under each major hypothesis.

1. Our first hypothesis was that a small representative subimage (and associated textures) of a pressure ridge could be used to evolve an individual. In a sense, we defined a pressure ridge to be a somewhat bright ridge feature in a patch. Our first implementation used an 8×8 pixel training set, which corresponded to 64 fitness cases. The results from these runs were unpublished: GP simply evolved individuals that essentially classified every pixel as background (i.e., matrix zeroing). In retrospect, these results were not surprising, since each 8×8 pixel subimage training set contained as few as 6-12 ridge points, and as many as 52-58 background points.

2. Our second hypothesis was that test points of roughly equal amounts of background and ridge points could be used to evolve an individual. We defined a ridge feature as a somewhat bright feature that is bracketed by somewhat darker features. Implementation and corresponding results are described in [Daida et al. 1995a]. The use of test points yielded two immediate benefits. First, an even ratio of ridge to background points discouraged the evolution of matrix-zeroing algorithms. Second, the small number of fitness cases reduced run times from CPU days to CPU hours (or minutes). The most significant shortcoming of this approach was in having too few fitness cases, which often allowed for individuals that did

not exhibit ridge finding behavior yet obtained perfect fitness scores within a few generations. The problem presented to GP was too "easy" and fitness was not closely enough linked to ridge finding behavior.

3. Our third hypothesis was to increase the number of fitness cases, as well as to include points that would be more difficult to classify. We subsequently defined a ridge feature to be a somewhat bright feature that is bracketed by somewhat darker features, in addition to specifying many of the other instances that were not ridges. This particular hypothesis did not involve a change in implementation. However, this strategy resulted in a breakdown of continuous ridge features into singlets and doublets. Apparently, over-specification of the fitness function created algorithms not robust enough to perform on larger image segments.

4. Our fourth hypothesis was to retain the idea of increasing the number of fitness cases, except this time it would be implemented along the idea of a default hierarchy. We defined a ridge as a somewhat bright and long feature that is bracketed by somewhat darker features. Emphasis was placed on finding the most clearly delineated ridges, with the more difficult ridges to be extracted by what was hoped to be an emergent property of the evolved solution. Implementation and corresponding results are described in this chapter. This hypothesis has proven to have been the most successful to date.

21.5.2 Sensitivities to Function-Set Implementation

For work in this chapter, we introduced two new operators P and M. These functions were introduced because of a result shown in Figure 21.6, which was evolved with the same fitness evaluation function, terminal, and function sets as were the results shown in Figure 21.4. The only noteworthy difference for the results shown in Figure 21.6 was the absence of operators P and M.

The algorithm that corresponds to Figure 21.6 was the individual $(- \text{Mean}_{5\times5}\ \text{Mean}_{3\times3})$. Over the course of several runs without P and M, the probability of finding this individual in any one run was high. This individual did score perfectly on all fitness cases. However, this individual still produced an unacceptable result, which showed the presence of pressure ridges even where none existed.

At first glance, the evolved algorithm looks similar to [Marr 1982]'s difference-of-Gaussians filter, which is a zero-crossings edge detector. Its behavior is also similar to Marr's filter. Note that on the right-hand side of Figure 21.6 in the area of the black stripe, there is a thin white stripe that extends the length of the image. This image artifact is consistent with the kind of artifact one would obtain with an edge detector.

Upon close inspection, however, our evolved "difference-of-means" algorithm was not quite what one would expect. We note that the histogram corresponding to the data shown in Figure 21.6 is not characteristic of a difference-of-Gaussians filter; there are humps at

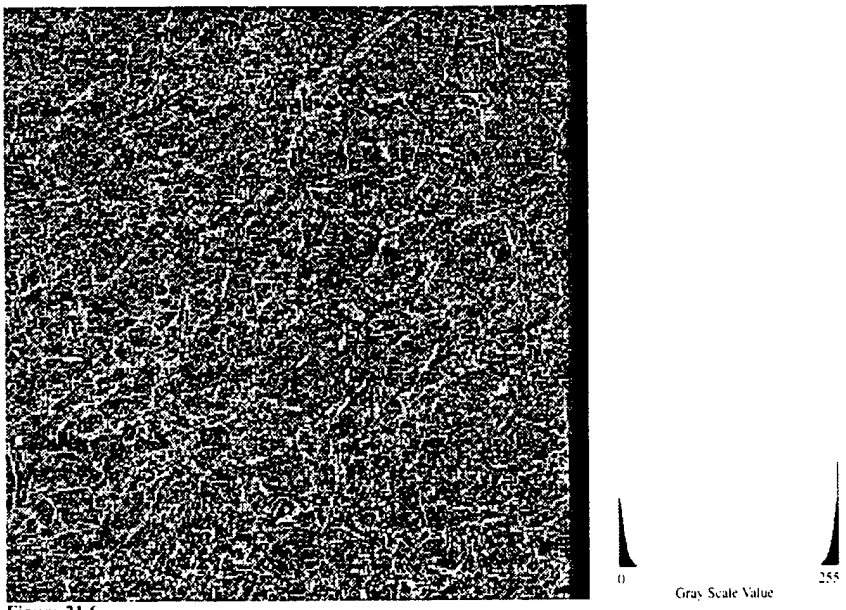


Figure 21.6
1024 × 1024 pixel ERS-1 SAR image (April 23, 1992) after filtering with the GP difference-of-means filter. Its gray-level histogram is also shown.

both ends of the gray scale. This histogram arose from our particular implementation of the arithmetic "–" operator. In particular, the use of mod 255 and unsigned byte type casting resulted in negative quantities showing up as numbers around 255. (What was novel about our "difference-of-means" filter was that GP exploited the nature of the "–" operator to create a nonlinear filter out of what should have been a linear one.) It was this unexpected histogram that prompted us to also include P and M in the function set.

21.5.3 Switch Filter Analysis

An analysis of the current best-of-runs individual shown in Figure 21.7—the switch filter—demonstrates how GP exploits seemingly benign idiosyncrasies that are built into a given function set. In particular, this individual has exploited the exception handling for the division operator to create a series of logic switches (hence the name "switch filter"). Furthermore, these logic switches were used to create two error-trapping devices that performed the task of exception handling between two imperfect ridge-finding subroutines.

The first error-trapping device prevented the division operator from returning an unde-

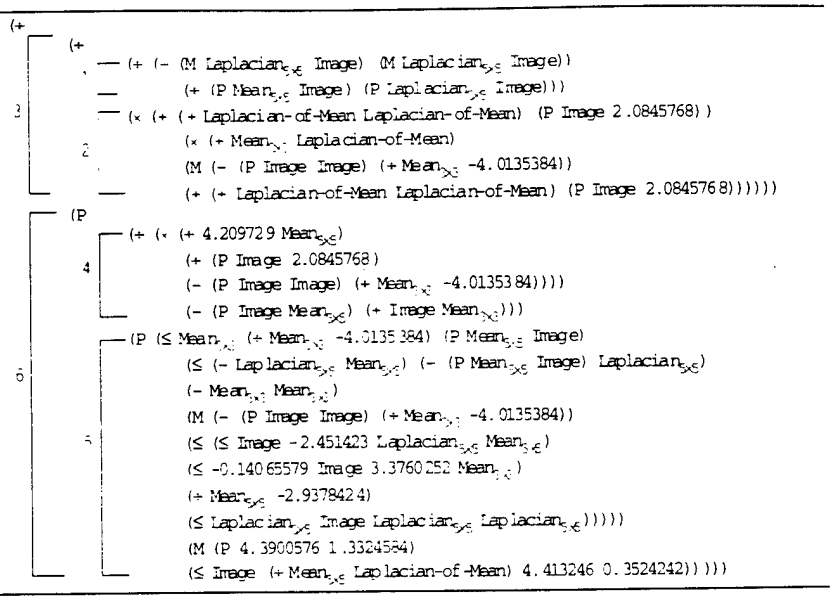


Figure 21.7
GP Switch-Filter Code.

fined result by setting division by zero to yield a value of one. In referring to Figure 21.7, we observed that Branch 2 zeroed-out all background points while Branch 1 always returned zero. With the numerator of the division operator at Node 2 set to zero, Node 2 produced a zero if its denominator was nonzero (ridge) and produced a one if its denominator was zero (background).

The second error trapping device prevented the division operator from returning an out-of-range value (i.e., greater than 255 or less than zero). The evolved code took advantage of this in the following way:

Branch 4 yielded nonzero values for ridge points (with only two exceptions among the 53 fitness cases) and zero values for every background point. Notably, Branch 5 almost zeroed the matrix, but instead left residual values of 10^{-3} to 10^{-4} for every fitness case. When Branches 4 and 5 were summed in Branch 6, the nonzero ridge points remained relatively unchanged, while significantly, zeroed background points acquired very small nonzero values.

The root-node divisor apparently handled exceptions and pushed the output values of the algorithm close to either 255 or zero. Background points took on their output value from evaluation of forms like (÷ 1.0 6.0×10^{-4}), which yielded 255. Ridge points took on their output value from evaluation of expressions like (÷ zero nonzero), which yielded zero.

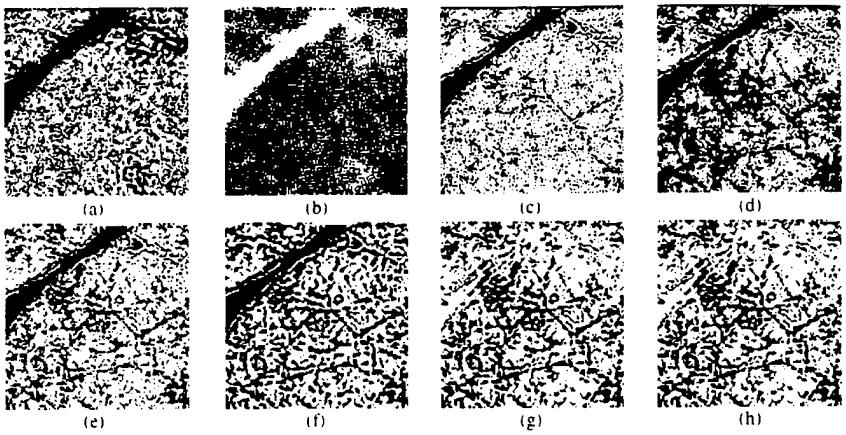


Figure 21.8
Evolution of GP Switch Filter. This series shows the best-of-generation results of filtering the 128 × 128 pixel ERS-1 SAR image shown in Figure 21.4. (a) Generation 0 (10 hits out of 15 possible test points). (b) Generation 1 (19 out of 24). (c) Generation 4 (31 out of 33). (d) Generation 7 (49 out of 53). (e) Generation 12 (51 out of 53). (f) Generation 18 (52 out of 53). (g) Generation 26 (53 out of 53). The best-of-generation individual did not change much after this, in part because there were no further fitness cases to consider. (h) Generation 28 (53 out of 53).

Ridge points misclassified by the subroutine represented by Branch 3 were corrected through division with a large number from Branch 6. For example, Branch 3 incorrectly classified fitness-case seven as a background point. The subroutine represented by Branch 6, however, would correctly classify this point as a ridge point. Evaluation at the root node, (÷ 1.0 65.9) yielded 0.0154, which has the net result of mediating the conflict between ridge-finding subroutines by correctly assigning the output for fitness-case seven a ridge value.

In effect, the root-node division operator acted as a final line of defense against misclassification. With two different sub-algorithms classifying each fitness case, it simply acted as a conflict moderator. For fitness-case seven, while the Branch 3 made an oversight, the Branch 6 sub-algorithm did not make the same mistake and adjusted for the error. While it was certain that a number of points on a 1024 × 1024 pixel images may be misclassified by both sub-algorithms, the odds of both failing simultaneously was much smaller than the odds of one or the other failing.

Figure 21.8 charts the evolution of the switch filter. In all instances shown in this figure, black corresponds to ridges; white, to background. Note that:

• Generation 0 yielded an individual that picks out some ridge features and some background features correctly.

• By Generation 1, an individual has adopted a strategy of using the division operator for a logic switch. Although the ridge classification contains a number of false positives, this individual correctly classified the dark diagonal structure as a non-ridge feature.

• By Generation 4, the switching strategy has become firmly adopted by many members of the population. An individual began to correctly pick out the most prominent ridge feature in the image.

• By Generation 7, the overall structure for most of the solution becomes evident. For several successive generations, the solution was continually refined.

• By Generation 26, an individual readopts a strategy for correctly classifying the dark diagonal structure as a non-ridge feature. Note that on one hand, the performance of this individual is fairly different from the performance of the individual shown in Generation 18. On the other hand, they are nearly similar on our fitness scale with an almost identical number of hits. This situation does consequently point out that a fitness metric is not necessarily a reliable indicator of performance.

21.5.4 Other Variations on Fitness

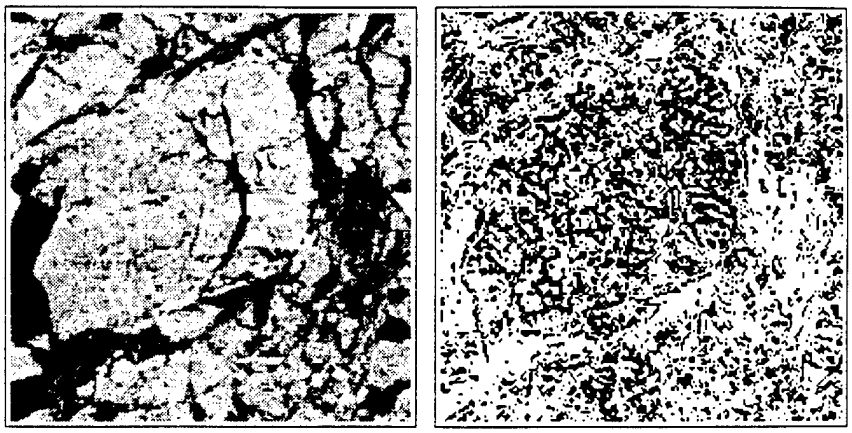
The described implementation of a dynamic fitness function based on default hierarchies represents but one instance. We have also considered but have not tried other means for triggering a change in fitness set during a run. These other implementations are subject for another paper.

21.6 Discussion: Domain

We leave an extended discussion of a geophysical interpretation based on data products like those shown in Figure 21.5 for another paper. However, we do have several comments about the results that have been shown in this chapter.

• The switch filter extracted both high- and low-contrast ridge features. This could be particularly important when a low-contrast ridge feature is located near a high-contrast ridge feature. In referring to Figure 21.9, we expect that a human operator would likely pick out the brightest ridges labeled "a," but miss the low-contrast ones labeled "b" because of the close proximity of the low-contrast features to the high-contrast ones. In contrast, the algorithm identified both low- and high-contrast features. Note that this behavior is an emergent property of the switch filter, since the filter was trained on only high-contrast features.

• The switch filter yielded textures that are based mostly on pressure ridge and rubble features, as opposed to edge features (as in Figure 21.6). These textures can be enhanced for visual inspection by smoothing. See Figure 21.10.



(a) (b)

Figure 21.9
Detail of Results from Switch Filter. (a) 200 × 200 pixel ERS-1 SAR image (April 23, 1992). Note pressure ridge features labeled with "a" are highly prominent, while ridge features labeled "b" are not. (b) From switch filter. The filter identifies both ridge features labeled "a" and "b."

• The textures that are present in Figure 21.5 are consistent with the geophysical explanation behind pressure ridges. The linear textures in areas where there is first-year ice are consistent with the expected compressive forces in those areas. The somewhat isotropic textures on multiyear ice are consistent with what is known about ice kinematics. Even though the process that generates leads and pressure ridges may be biased towards a certain directions, floes of multiyear ice continually rotate, which eventually subject those floes to ridge formations in all directions.

21.7 Conclusions

This chapter demonstrated how the genetic programming paradigm can apply to image processing in geoscience and remote sensing. A problem for which no known solution was studied, which involved extracting low-contrast curvilinear features from SAR imagery. This chapter has shown that the genetic programming paradigm can be used in the discovery process in deriving an appropriate spatial classifier.

We note that our overall approach in using test points in the context of dynamic training sets is generalizable to other similar problems that involve extracting features from multiband image data (by replacing data in multiple textural channels with multiple spectral

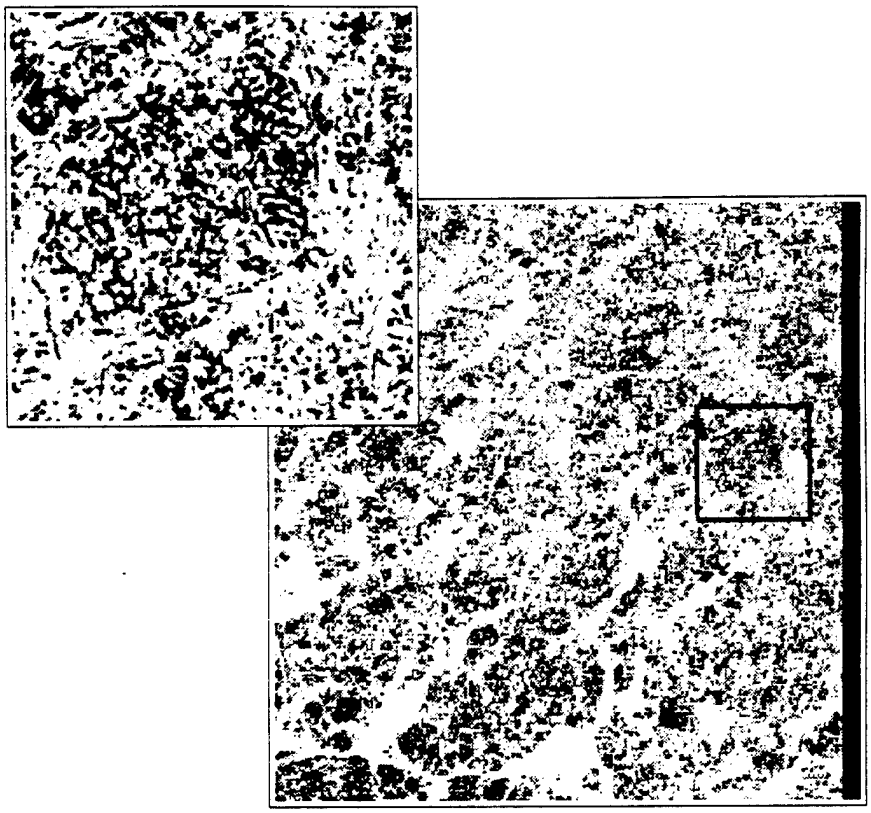


Figure 21.10
Smoothed Detail of Results from Switch Filter. Smoothing enhances visual perception of texture. 1024 × 1024 pixel ERS-1 SAR image (April 23, 1992). Inset: enlargement of 200 × 200 pixel boxed area.

channels). This approach should be generalizable since the problem-specific characterization has been made with only a few assumptions about image content.

Acknowledgments

This research has been partially funded with grants from the Office of Naval Research, the Naval Research Laboratory (Stennis) and the Space Physics Research Laboratory (U-M). We gratefully acknowledge the following people: Robert Onstott (Environmental Research

Institute of Michigan) for ERS-1 image data; Florence Fetterer (NRL), Drew Rothrock, Harry Stern (University of Washington), Robert Shuchman, and Fred Tanis (ERIM) for our discussions on pressure ridges; Ramesh Jain (University of California, San Diego), Elliot Soloway, Elke Rundensteiner, and Paul Hays (U-M) for our discussions on task analysis for scientific computation; Mark Guzdial (Georgia Institute of Technology), for scaffolding by computer; Peter Angeline (Loral Federal Systems), Kenneth Kinnear (Adaptive Computing Technology), Marc Schoenauer (Ecole Polytechnique), William Punch (Michigan State University) and Sandra Daida, for editorial support and technical review; Mark Sakala (U-M), for his timely computer work; and last but definitely not least, Rick Riolo (U-M) and John Koza (Stanford), for our discussions on GP.

Bibliography

Barrera, J., E.S.C. da Silva, & G.J.F. Banon (1994). "Automatic programming of binary morphological machines." *Image Algebra and Morphological Image Processing V: Proceedings of the SPIE*, **2300**, pp. 229–240.

Burns, B.A. & A. Wegener (1988). "SAR image statistics related to atmospheric drag over sea ice." *Proceedings of the International Geoscience and Remote Sensing Symposium*, ESA Publications, ESA SP-284, pp. 409–412.

Brown, A. & A. Palincsar (1989). "Guided cooperative learning and individual knowledge acquisition." *Knowing, Learning and Instruction: Essays in Honor of Robert Glaser*, L.B. Resnick (ed.), Hillsdale: Lawrence Erlbaum Associates, pp. 393–451.

Chang, S.K., et al. (1990), ed., *Visual Languages*, Plenum Press, 1986, second edition 1990.

Chen, C.-H. (1992). "Automatic vision programming." *CVGIP: Image Understanding*, **55**:2, pp. 170–83.

Chien, S. (1994). "Using AI planning techniques to automatically generate image processing procedures: A preliminary report." *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp. 219-224.

Congalton, R.G. (1991). "A review of assessing the accuracy of classifications of remotely sensed data." *Remote Sensing of the Environment*, **37**:1, pp. 35-36.

Daida, J.M., J.D. Hommes, S.J. Ross, & J.F. Vesecky (1995a). "Extracting curvilinear features from synthetic aperture radar images of arctic ice: Algorithm discovery using the genetic programming paradigm." *Proceedings of the 1995 International Geoscience and Remote Sensing Symposium: Quantitative Remote Sensing for Science and Applications*, IEEE Press, pp. 1415–1417.

Daida, J.M., A Freeman, & R.G. Onstott (1995b). "Evaluation of a hybrid symbiotic system on segmenting SAR imagery." *Proceedings of the 1995 International Geoscience and Remote Sensing Symposium: Quantitative Remote Sensing for Science and Applications*, IEEE Press, pp. 673–675.

Gineris, D.J. & F.M. Fetterer (1993). *The Joint Ice Center SAR Workstation: Algorithm Evaluation*, Memorandum Report 7019, Stennis Space Center, Naval Research Laboratory Detachment, March.

Goldberg, D.E. (1989), *Genetic Algorithms in Search, Optimization, and Machine Learning*, Reading: Addison-Wesley.

Gow, A.J. & W.B. Tucker III (1990). "Sea ice in the polar regions." *Polar Oceanography, Part A: Physical Science*, W.O. Smith, Jr. (ed), San Diego: Academic Press, Inc., pp. 47–122.

Guzdial, M. (1995). "Software-realized scaffolding to facilitate programming for science learning." *Interactive Learning Environments*, **4**:1, pp. 1-44.

Haralick, R.M. & L.G. Shapiro (1992). *Computer and Robot Vision: Volume I*, Reading: Addison-Wesley Publishing Company, Inc.

Holland, J.H. (1992). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Cambridge: MIT Press 1992. First edition 1975. The University of Michigan.

Holland, J.H., K.J. Holyoak, R.E. Nisbett, & P.R. Thagard (1986). *Induction: Processes of Inference, Learning, and Discovery*. Cambridge: MIT Press.

Hsu, R.C. & S.S. Alexander (1994). "A neural network approach to seismic event identification using reference seismic images." *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*. pp. 2108–13.

Koch, M.W. & M.M. Moya (1994). "Feature discovery in gray level imagery for one-class object recognition." Proceedings of the IEEE Int Conference on Neural Networks. IEEE Press. pp. 2979–84.

Konstatinides, K. & J. Rasure (1994). "The Khoros software development environment for image and signal processing." *IEEE Transactions Image Processing*, **3**:3. pp. 243–252.

Koza, J.R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge: MIT Press.

Koza, J.R. (1994). *Genetic Programming II: Automatic Discovery of Reusable Programs*. Cambridge: MIT Press.

Lillesand, T.M. & R.W. Kiefer (1987). *Remote Sensing and Image Interpretation*. Second Edition. New York: John Wiley and Sons.

Lunetta, R.S., R.G. Congalton, L.K. Fenstermaker, J.R. Jensen, K.C. McGwire, & L.R. Tinney (1991). "Remote sensing and geographic information system data integration: error sources and research issues." *Photogrammetric Engineering and Remote Sensing*, **57**:6. pp. 677-87.

Marr, D. (1982). *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. New York: W.H. Freeman and Company.

Matwin, S., D. Charlebois, & D.G. Goodenough (1995). "Training agents in a complex environment." *Proceedings of the 11th Conference on Artificial Intelligence for Applications*. IEEE Computer Society Press. pp. 94-100.

Mellor, M. (1986). "Mechanical behavior of sea ice." *The Geophysics of Sea Ice*. N. Untersteiner (ed.). New York: Plenum Press. pp. 165–281.

Merrill, D.C. & B.J. Reisner (1993). "Scaffolding the acquisition of complex skills with reasoning-congruent learning environment." *Workshop in Graphical Representations, Reasoning and Communication: World Conference on Artificial Intelligence in Education*. The University of Edinburgh. pp. 9–16.

Nguyen, T. & T. Huang (1994). "Evolvable 3D modeling for model-based object recognition systems." *Advances in Genetic Programming*. K.E. Kinnear, Jr. (ed.). Cambridge: MIT Press. pp. 459–475.

Openshaw, S. (1995). "Developing automated and smart spatial pattern exploration tools for geographical information system applications." *Statistician*, **44**:1. pp. 3–16.

Rasure, J. & C. Williams (1991). "An integrated data flow visual language and software development environment." *Journal of Visual Languages and Computing*. pp. 217–246.

Samadani, R. & J.F. Vesecky (1990). "Finding curvilinear features in speckled images." *IEEE Transactions Geoscience and Remote Sensing*, **28**:4. pp. 669–673.

Tackett, W.A. (1993). "Genetic programming for feature discovery and image discrimination." *Proceedings of the Second International Conference on AI Planning Systems*. pp. 303-309.

Vesecky, J.F., M.P. Smith, & R. Samadani (1990). "Extraction of lead and ridge characteristics from SAR images of sea ice." *IEEE Transactions Geoscience Remote Sensing*, **28**:4. pp. 740-744.

Vogt, R.C. (1989). *Automatic Generation of Morphological Set Recognition*. New York: Springer-Verlag.

Vygotsky, L. (1978). *Mind in Society*. Cambridge: Cambridge University Press.

Wood, D., J.S. Bruner, & G. Ross (1975). "The role of tutoring in problem solving." *Journal of Child Psychology and Psychiatry*, **17**. pp. 89–100.

Paper 3

Daida, J.M., R.G. Onstott, T.F. Bersano-Begey, S.J. Ross, and J.F. Vesecky, Ice Roughness Classification and ERS SAR Imagery of Arctic Sea Ice: Evaluation of Feature-Extraction Algorithms by Genetic Programming, 1520-1522, **Proceedings of the 1996 International Geoscience and Remote Sensing Symposium**, Washington: IEEE Press (1996b).

To appear in the *Proceedings of the 1996 International Geoscience and Remote Sensing Symposium: Remote Sensing for a Sustainable Future*, Washington: IEEE Press, 1996.

Ice Roughness Classification and ERS SAR Imagery of Arctic Sea Ice: Evaluation of Feature-Extraction Algorithms by Genetic Programming

Jason M. Daida*, Robert G. Onstott**, Tommaso F. Bersano-Begey*, Steven J. Ross*, & John F. Vesecky*
*The University of Michigan, Department of Atmospheric, Oceanic and Space Sciences,
2455 Hayward Avenue, Ann Arbor, Michigan, USA 48109-2143, (313) 747-4581 (work), (313) 764-5137 (fax), daida@eecs.umich.edu
**Environmental Research Institute of Michigan, Center for Earth Sciences, Advanced Concepts Division
P.O. Box 134001, Ann Arbor, Michigan 48113-4001

Abstract — This paper describes a validation of accuracy associated with a recent algorithm that has been designed to extract ridge and rubble features from multiyear ice. Results show that the algorithm performs well with low-resolution ERS SAR data products.

1. INTRODUCTION

Roughness in the polar ice cover—like meso-scale features of pressure ridges and rubble fields—is of significant geophysical interest. Pressure ridges and rubble fields help to transfer kinetic energy from meteorological systems to the ice cover. Pressure ridges can significantly increase sea-ice drag coefficients, which subsequently affect sea-ice movement and deformation. Ridges and rubble fields are also of interest because they account for a large portion of the total ice mass.

In ERS synthetic aperture radar (SAR) imagery, pressure ridges commonly appear as filamentary, curvilinear features of variable width. These features have radar backscatter signatures that differ only slightly from those of non-ridged multiyear-ice; pressure ridges subsequently appear as mostly low-contrast features. Rubble fields often form when sea ice undergoes multiple ridging events in the same geographic region. Not surprisingly, rubble fields have backscatter signatures that are similar to that of pressure ridges, except that rubble fields may have shapes ranging from consolidated blobs to interlaced networks of curvilinear features.

The difficulty in extracting such features has been noted in work such as [1]. The problem has been considered untenable for standard image processing algorithms for a variety of reasons. Such reasons include low signal-to-noise ratios, arbitrariness of feature shapes, and radar cross-sections that change depending on the orientation of a feature. The problem, however, has not been considered impossible, since researchers have been able to link the roughness caused by sea ice deformation (like ridges and rubble fields) with ERS SAR backscatter. [4]

This paper evaluates an ice-roughness algorithm that we have developed over the past year. The next section (2) briefly discusses our algorithm. Section 3 describes our procedure for evaluating this algorithm, which involves validation of a derived data product from this algorithm with an area that has ground truth. Section 4 presents and discusses our results. Section 5 summarizes our major conclusions.

2. ALGORITHM NOTES

Our algorithm has been developed by using a relatively new procedure in computer-assisted software design. This procedure, which uses genetic programming, has been developed to help a user to focus more on the problem at hand and less on programming detail. Another paper in this conference highlights some of the salient characteristics of our procedure [2].

We have designed our algorithm to extract ridge and rubble features in multiyear ice. It has been developed for use with low-resolution (ERS) SAR data products, partly because we desired meso-scale distributions and partly because we wanted to track temporal changes. For more information on the development of this particular algorithm (called a switch filter), see [3].

3. PROCEDURE

The image that we have chosen for validation is part of a larger series of temporal ERS-1 SAR data that we have analyzed. The series, which begins in August 1991 and ends in July 1992, describes the synoptic coverage of an area in the Beaufort Sea gyre (roughly 72°N, 140°W). The particular area and dates of coverage overlap with the Lead Processes Experiment (LEADEX) in 1992, which featured both in-situ and ERS-1 observations around a chosen floe.

Figure 1a shows the low-resolution ERS SAR image taken 29 March 1992, while Figure 1b shows the corresponding data product derived with the ice-roughness algorithm. We note that the algorithm was developed using data from a different image (23 April 1992)—the 29 March data is entirely out-of-sample. The boxed area shown in both these figures corresponds to the ice classification map shown in Figure 1c. Classifications for this map were based on ground observations from the LEADEX base camp.

To validate the derived data product, we used a ridge and rubble map that was manually obtained from the high-resolution (nominally 12 m resolution) ERS SAR data product for the same day and area. To ensure accuracy, we limited the extent of this ridge and rubble map to the immediate area (~ one km) around the base camp. The map was verified by personnel present at the base camp at the time the image was taken (i.e., R. Onstott). The boxed area in Figure 1c shows theextent of the manually derived ridge and rubble map.

4. RESULTS AND DISCUSSION

Figures 2a – d show our results. Figure 2a depicts the high-resolution subimage that was used to create the manually derived ridge and rubble map. The image has been enhanced for publication to highlight those features, which show as light gray pixels on a gray background. (Gray generally corresponds to multiyear ice, while dark gray generally corresponds to first-year ice.) The three bright collinear dots in the center of this figure correspond to corner reflectors placed on first year ice. (A fourth dot—another corner reflector—is also visible, but on multiyear ice.)

Figure 2b shows the manually derived ridge and rubble map. (Black denotes ridge and rubble features in multiyear ice, or extreme ridging in first year ice. Note that the three dots have been retained for comparison.)

Figure 2c shows the results from the ice-roughness algorithm.

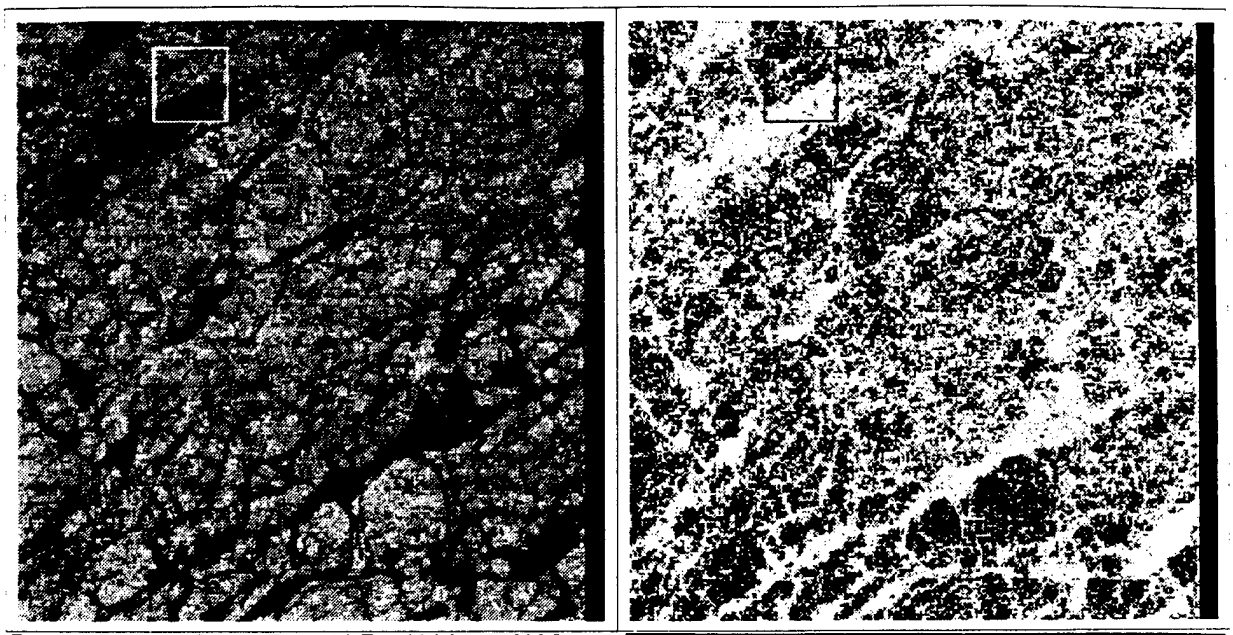


Figure 1. Data and Ground Truth. (a) Top. 29 March 1992 Image (1024 × 1024 pixels) © ESA 1992. (b). Upper right. Derived Data Product. (c) Lower Right. Map of Ground Truth.

Note that the pixels are noticeably larger than those shown in Figures 2a and 2b. This is expected, since the ice-roughness algorithm works on low-resolution data products. (Black and dark grays denote ridge and rubble features.)

Figure 2d shows the results of overlaying the results from the ice-roughness algorithm on the manually derived ridge and rubble map. (Black indicates a high degree of correlation for ridge and rubble features, while white indicates a high degree of correlation for smooth features. Gray without any interior black denotes areas of possible conflict.)

The results show excellent correspondence between the manually derived map and the data product from the ice-roughness algorithm. Much of the identified ridge and rubble features in multiyear ice have been correctly classified in the data product. Tolerance accuracies in the data product are better than ±100 m (± 1 pixel) of a ridge or rubble feature in the high-resolution map.

We note that the data product shows a correct classification of ridge features in an area just below the three collinear dots in Figure 2a. Ground truth corresponding to this area indicates an area of old pressure ridges—worn and smoothed. Radar backscatter signatures corresponding to ridges like these are not much different from non-ridged multiyear ice; such features are difficult to classify.

The ice-roughness algorithm does seem to identify ridges and rubble features regardless of whether such features yield strong or weak signatures in contrast to the mean backscatter values of multiyear ice. If this is the case, such an attribute would help to desensitize the algorithm from ridge orientation effects on radar backscatter.

We further note that the algorithm has classified a series of pixels in the lower right corner of Figure 2c as ridge or rubble features, even though such pixels correspond to areas of first-year ice. The algorithm apparently identifies a few, but not all the ridge and rubble features in first-year ice. These features do appear in the image data but not in Figure 2a. (As mentioned earlier, we enhanced Figure 2a to show, for publication, ridge and rubble features in multiyear, not first-year ice.) We have found that the few first-year ridge and rubble features that have been identified do show a high correspondence with major stress and deformation patterns in first-year ice. (See [3].)

4. CONCLUSIONS

This paper has evaluated the performance of an ice-roughness algorithm that was developed using a relatively new procedure in

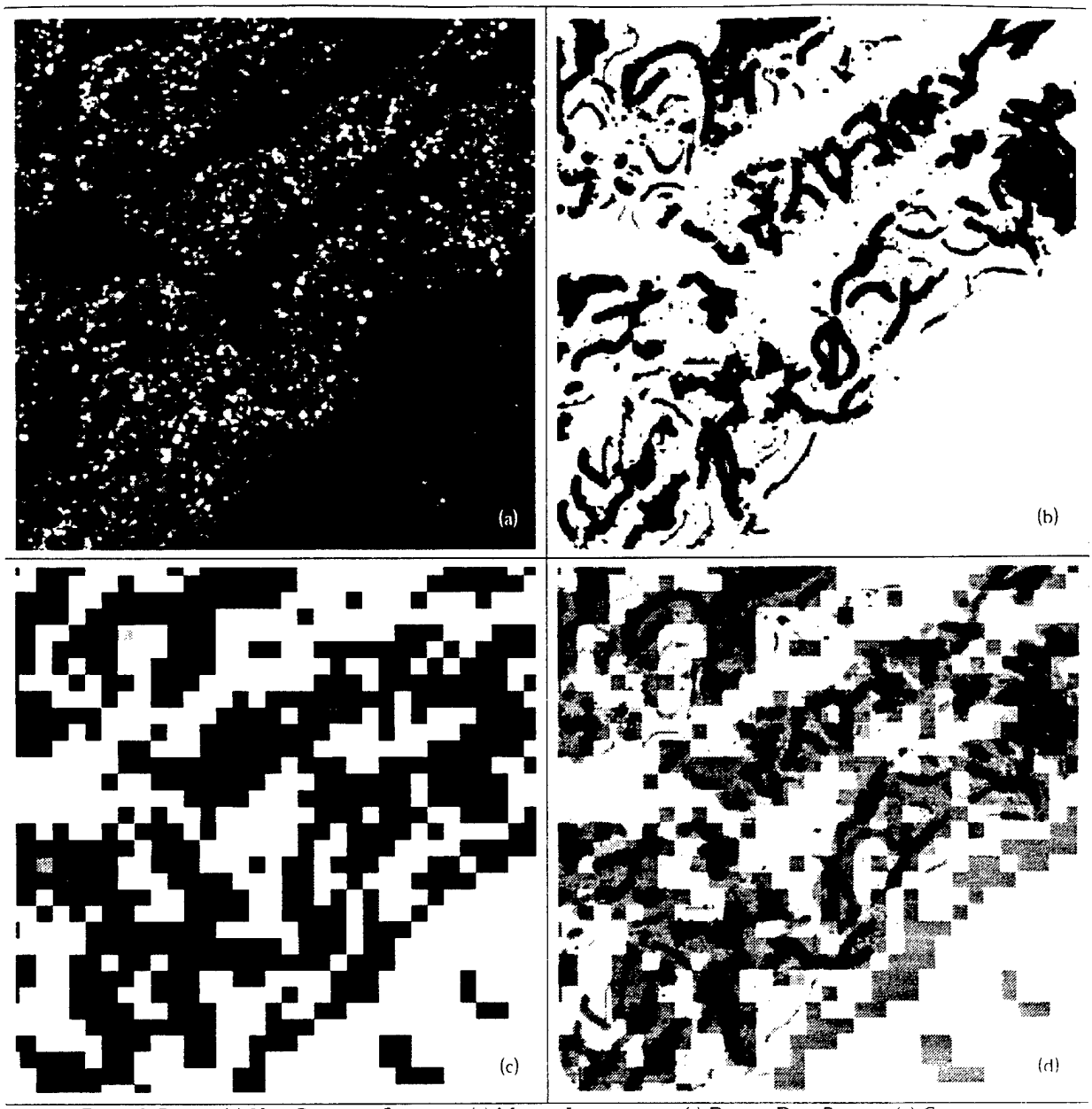


Figure 2. Results. (a) High-Resolution Subimage. (b) Manual Interpretation. (c) Derived Data Product. (d) Comparison.

computer-assisted software design. To evaluate this algorithm, we used ERS SAR image data that coincides with *in-situ* observations obtained during LEADEX '92. The results have shown excellent agreement between the derived data product and a manually interpreted ERS SAR data product. The algorithm has been shown to extract features corresponding to ridges and rubble fields in multiyear ice. We have suggested that the algorithm does extract enough of the ridge and rubble features in first-year ice to show gross deformation patterns. ■

BIBLIOGRAPHY

[1] Burns, B.A., "SAR Image Statistics Related to Atmospheric Drag Over Sea Ice," *IEEE Trans. GRS*, **28**:2, p. 158–65.

[2] Daida, J.M., T.F. Bersano-Begey, S.J. Ross, and J.F. Vesecky, "Evolving Feature-Extraction Algorithms: Adapting Genetic Programming for Image Analysis in Geoscience and Remote Sensing," *Proceedings of IGARSS '96*, IEEE Press. In Press.

[3] Daida, J.M., J.D. Hommes, T.F. Bersano-Begey, S.J. Ross, and J.F. Vesecky, "Algorithm Discovery Using the Genetic Programming Paradigm: Extracting Low-Contrast Curvilinear Features from SAR Images of Arctic Ice," *Advances in Genetic Programming II*, P. Angeline and K. Kinnear (ed.). Cambridge: The MIT Press, 1996. In Press.

[4] Onstott, R.G., D. Miller, and R.A. Shuchman, "Study of the Relationship Between the Scale of Sea Ice Deformation and Radar Backscatter Intensity Using ERS-1 SAR," *Proceedings of IGARSS '95*, IEEE Press, pp. 419–421.

Paper 4

Daida, J.M., T.F. Bersano-Begey, S.J. Ross, and J.F. Vesecky, "Evolving Feature-Extraction Algorithms: Adapting Genetic Programming for Image Analysis in Geoscience and Remote Sensing, 2077-2079, **Proceedings of the 1996 International Geoscience and Remote Sensing Symposium**, Washington: IEEE Press (1996c).

To appear in the *Proceedings of the 1996 International Geoscience and Remote Sensing Symposium: Remote Sensing for a Sustainable Future*, Washington: IEEE Press, 1996.

Evolving Feature-Extraction Algorithms: Adapting Genetic Programming for Image Analysis in Geoscience and Remote Sensing

Jason M. Daida*, Tommaso F. Bersano-Begey*, Steven J. Ross*, and John F. Vesecky**

*The University of Michigan, Artificial Intelligence Laboratory and the Space Physics Research Laboratory
2455 Hayward Avenue, Ann Arbor, Michigan 48109-2143
(313) 747-4581 FAX (313) 764-5137 EMAIL: daida@eecs.umich.edu
**The University of Michigan, Department of Atmospheric, Oceanic and Space Sciences
2455 Hayward Avenue, Ann Arbor, Michigan 48109-2143

Abstract—This paper discusses a relatively new procedure in the computer-assisted design of pattern-extraction algorithms. The procedure involves the adaptation of genetic programming, a recent technique that has been used for automatic programming, for image processing and analysis. This paper summarizes several of the measures we have taken to develop two prototype systems that help a user to design pattern-extraction algorithms.

1. INTRODUCTION

The task of extracting patterns from remotely sensed imagery can be difficult, particularly if increasingly novel patterns need extraction or when using data from new imaging technologies. In such cases, off-the-shelf software—e.g., geographic image processing systems or image analysis software libraries—may be able to meet a part, but not all the specifications for a pattern-extraction problem at hand. To fulfill such specifications, the best software tool to use may end up being code that has been specifically designed for a particular problem.

To assist in the development of problem-specific pattern-extraction tools, there has existed a range of options from which one could choose. On one hand, one could choose to program such tools from scratch, usually in a high-level programming language. On the other hand, one could also program in an even higher-level macro-language (e.g., many geographic image processing systems or scientific visualization systems often feature a macro- or a scripting-language capability). In either case, success of custom-tailored, pattern-extraction tools resides not only in one's ability to design a procedure with which to extract patterns, but also in one's ability to program.

The overhead associated with programming can be substantial and often include such tasks as specifying software, programming, debugging, and testing, as well as contending with learning curves associated with new languages, new software libraries, or new programming tools. To reduce this overhead would be a desirable objective, since programming often represents only a means (albeit a time-consuming one) for delivering what is needed—namely, a tool for extracting a particular pattern.

In part to address this objective, previous work has included investigations of strong and weak methods in artificial intelligence. The terms "strong" and "weak" refer not to a measure of a method's performance, but to the amount of knowledge about a given problem that a method requires. An example of a "strong" method is an expert system, which consists of a substantial portion of code that is problem-specific and a smaller portion of code that is generic. An example of a weak method is a neural network, which consists of a substantial portion of code that is largely generic, and a smaller part that is problem-specific. Either of these methods can still involve a fair amount of programming, although one could argue that these methods can reduce programming overhead. (In particular, by making programs more adaptable, more robust, and more intelligent, less programming would be required on the part of a user.)

In the past several years, there have been new techniques in weak methods that show additional promise in reducing programming overhead. This paper describes one of these techniques and highlights how this particular technique can be used for creating problem-specific pattern-extraction algorithms for use with remotely sensed image data.

2. GENETIC PROGRAMMING

This paper specifically discusses our use of genetic programming, an unsupervised technique that generates computer code. Originally introduced in 1989 by Koza [5], genetic programming has since been applied in a variety of domains, ranging from molecular biology to robotics to economics. Genetic programming has been shown to solve a number of problems that have served as benchmarks for neural nets [4, 6]. It has also been shown to solve problems in which neural nets might not be the most appropriate technology to use (e.g., symbolic regression) [6].

Genetic programming does have several attributes that potentially justify its use with remotely sensed image data. First, it has been shown to work with imperfect or incomplete problem specifications, while still yielding reasonable solutions [2]. Second, it has been able to produce extremely robust classification code that yields accuracies that are comparable to that which has been obtained with manually produced algorithms [4]. Third, it has been able to work with linear or nonlinear problems with little or no changes to its core (generic) portion [6, 7]. Fourth, it has been shown to work with a wide range of data types, including multi-channel image data [2]. Fifth, it has been able to output several types of code ranging from macros (which would require some type of interpreter) [6] to assembler [9]. Sixth, it has been used to automatically produce code with features such as subroutines, registers, and iterative loops [8]. Seventh, it has been routinely used to manipulate both symbolic and numeric data [6, 7].

Basic genetic programming can be briefly described as follows. A genetic programming run starts with a set of

randomly created programs that have been generated from the components that a user has supplied. We presume that somewhere in this random collection of programs are the building blocks necessary for the desired solution. The trick, of course, is to sift through this random code in order to synthesize a desired solution from these building blocks. To implement this trick, genetic programming uses two operations that are modeled after biological processes: natural selection and genetic crossover. In this case, natural selection means that only the most fit individuals reproduce and have offspring. In terms of an operator, natural selection means that these random programs need to be ranked by performance. Ranking is accomplished by means of a fitness function, which has been specified by a user before a run. A fitness function is supplied as a subroutine that tests for how close a program comes to a known result given a known set of inputs. Reproduction and the bearing of offspring refer to the biological process of how offspring are the genetic composite of both parents—i.e., through genetic crossover. In terms of an operator, crossover means that a portion of code from one program is replaced with a portion of code from another program. The resulting composite program is analogous to an "offspring." Crossover occurs mostly among the fittest programs (the pairing of prospective parents is stochastic, but probabilities favor the fittest programs) and continues until a new population of offspring is attained. The operations corresponding to natural selection and crossover are then repeated for this new population. A genetic programming run continues for subsequent populations until a candidate program obtains the best score allowable under the user-supplied fitness function.

Detailed descriptions of genetic programming can be found in [6, 7].

3. PROBLEMS

In spite of its apparent advantages and benefits, genetic programming has only recently appeared in the geoscience and remote sensing literature. Part of this relatively late appearance can be traced to two difficulties. One difficulty involves computational overhead. In Section 2, we noted that basic genetic programming involves the creation and test of many individual candidate programs. A typical run in genetic program may involve the creation and test of several thousand of such candidate programs. For numerous problems previously addressed by genetic programming, testing of each candidate solution is relatively quick and inexpensive (e.g., each program uses several tens of kilobytes of memory and executes in under a minute). For problems involving remotely sensed imagery, candidate solutions could easily use major blocks of memory and minutes to hours of CPU time.

The other difficulty is that the canonical genetic programming system, which has been freely available for some time, is in LISP. (The code for the canonical genetic programming system can be obtained at ftp://ftp.io.com/pub/genetic-programming.) While LISP is arguably the most intuitive language for genetic programming to use, LISP has a few idiosyncrasies that can hinder processing remotely sensed imagery.

4. ADAPTATIONS

This section summarizes our experiences to date in adapting genetic programming for processing remotely sensed imagery [1, 2, 3]. Our particular experience involves the extraction of low-contrast ridge and rubble patterns in low-resolution ERS synthetic aperture radar images of arctic sea ice. (See [3] in this conference proceedings.) Our problem has been relatively typical of those that involve multiple channels of image data (in our case, several textural channels).

To solve our problem, we have implemented six adaptations to genetic programming. The first three have helped in reducing computation time (for a LISP version of our system) to several CPU hours on a Sun SPARCStation 20 workstation. A fourth adaptation has enabled the LISP version of our system to process megabyte images. A fifth adaptation has helped to further reduce computation time to tens of minutes. A sixth adaptation has been proven necessary when using our methods for directly processing images with genetic programming.

4.1 Preprocessing

We preprocessed as much of the data as was possible to reduce genetic programming run time. For our problem, this has meant that the texture measures were computed beforehand. In particular, our problem was one in which it was possible to use texture channels, where each channel represents a different filtered version of the image data to be processed. (Two of the channels we specified corresponded to layers in an image pyramid: mean images filtered with kernel sizes of 3×3 and 5×5, respectively. The other two channels corresponded to edge detection: a Laplacian image of kernel size 5×5 and a Laplacian, 5×5, of a mean, 3×3, image.) In a sense, what was left for genetic programming to do was to formulate an algorithm (a rule set) that governs how the data in each channel was to be combined with the others.

Note that a complete algorithm produced by genetic programming would then consist not only of a rule set, but would also include the subroutines implicit with each filtered channel. While this may seem obvious, there may be unintentional side effects if one simply converted the algorithm produced by genetic programming to a standalone application and then used a different software implementation to generate each filtered channel.

4.2 Test Points

We used test points in an image that have been manually interpreted to serve as programming benchmarks. Each candidate algorithm produced in genetic programming is executed with these test points to assess that algorithm's accuracy in extracting a desired pattern (i.e., in the context of a fitness function). The number of test points that need to be provided does not have to be large. For our problem, the best algorithm that we have obtained to process full-sized low-resolution ERS SAR data products was developed using only 53 test points.

Note that we used test points, rather than subimage patches, to serves as "training sets" for genetic programming. We have found that the use of subimage patches was too constraining, did not help, and has even hindered processing times. See [2].

4.3 Dynamic Fitness

Instead of requiring all candidate algorithms to use a fixed standard, we opted to use a sliding standard. By that we mean the number of test points needing to be solved at the outset of a genetic programming run are fewer than the number of test points needing to be solved towards the end of that run. We have found this technique to result in better quality algorithms than without this technique, when genetic programming needs to solve for the entire test point set at the outset. See [1, 2] for implementation details.

4.4 Chunking

LISP requires a fairly sizable amount of processing overhead, which negatively affects the size of an image that can be processed at any one time. As a workaround intended mostly for LISP systems, we chunked low-resolution ERS SAR image data into smaller subimages, processed the subimages, then integrated the processed subimages to obtained a whole derived data product. The nature of the operators that we used for genetic programming allowed for seamless integration of subimages.

Our first prototype was built around the canonical genetic programming kernel, which means that our prototype system was implemented in LISP.

4.5 C-Language Port

As well suited for LISP as genetic programming is, there have been several reasons that have prompted us to design our second prototype in C: larger array sizes, faster run times, broader user community, wider range of programming flavors and tools. We have developed our second prototype around a recently introduced C-language version of genetic programming. (The code for the C-language version of the genetic programming kernel is available at the following URL: http://isl.cps.msu.edu/GA/software/lil-gp/) Early tests have shown that our second prototype runs about an order of magnitude faster than our LISP version.

4.6 Scaffolding

More often times than not, in cases where custom-designed code is needed, one is not able to start with an exact specification of a pattern. There often exists an uncertainty on what does, indeed, constitute the desired pattern. This uncertainty can show up as inappropriately interpreted test points, or even as inappropriately chosen test points. Several iterations of trying and interpreting different test points are usually the norm.

For this and other reasons described in [1, 2], we have designed our system with the deliberate intent to involve the user. In a sense, we have designed our system so that genetic programming facilitates the testing of a user's hypotheses on what a desired pattern should be. The user learns about specifying a desired pattern in a consistent manner, while the computer assumes the role of a human programmer, who would logically extend a user's specification into code. This cooperative relationship has been referred in the education and technology literature as scaffolding.

5. CONCLUSIONS

This paper has highlighted some of the adaptations used to incorporate genetic programming in the design of algorithms that extract features from remotely sensed images. In particular, this paper has summarized six adaptations: preprocessing, test points, dynamic fitness, chunking (for LISP versions), C-language port, and scaffolding. Although many applications have employed genetic programming as an unsupervised method, we have integrated genetic programming as part of an interactive system that serves as a tool for computer-assisted algorithm design. The overall system has been developed to help a user to focus more on the problem at hand and less on programming detail.

Additional references on genetic programming include [6, 7]. One can also obtain information and software about genetic programming at the following URL addresses:

- http://www.cs.ucl.ac.uk/research/genprog/
- ftp://ftp.io.com/pub/genetic-programming

ACKNOWLEDGMENTS

This research has been partially funded with grants from the Office of Naval Research, the Naval Research Laboratory (Stennis) and the Space Physics Research Laboratory (U-M). We gratefully acknowledge the following individuals for their invaluable assistance: I. Kristo, S. Daida, J. Hommes, J. Koza, R. Onstott, R. Riolo, E. Soloway, and A. Wu.

BIBLIOGRAPHY

[1] Daida, J.M., T.F. Bersano-Begey, S.J. Ross, and J.F. Vesecky. "Computer-Assisted Design of Image Analysis Algorithms: Dynamic and Static Fitness Evaluations in a Scaffolded Environment." Submitted to *Genetic Programming '96*.

[2] Daida, J.M., J.D. Hommes, T.F. Bersano-Begey, S.J. Ross, and J.F. Vesecky. "Algorithm Discovery Using the Genetic Programming Paradigm: Extracting Low-Contrast Curvilinear Features from SAR Images of Arctic Ice." *Advances in Genetic Programming II*. P. Angeline and K. Kinnear, Jr., (ed.). Cambridge: The MIT Press, 1996. In Press.

[3] Daida, J.M., R.G. Onstott, T.F. Bersano-Begey, S.J. Ross, and J.F. Vesecky. "Ice Roughness Classification and ERS SAR Imagery of Arctic Sea Ice: Evaluation of Feature-Extraction Algorithms by Genetic Programming." *Proceedings of IGARSS '96*. IEEE Press. In Press.

[4] Francone, F.D., P. Nordin, B. Banzhaf. "Benchmarking the Real-World Generalization Capabilities of an Advanced, Compiling Genetic Programming System Using Sparse Data Sets." Submitted to *Genetic Programming '96*.

[5] Koza, J.R. "Hierarchical Genetic Algorithms Operating on Populations of Computer Programs." *Proceedings of the 11th International Joint Conference on Artificial Intelligence*. Volume 1., Morgan Kaufmann.

[6] Koza, J.R., *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge: The MIT Press, 1992.

[7] Koza, J.R., *Genetic Programming II: Automatic Discovery of Reusable Programs*. Cambridge: The MIT Press, 1994.

[8] Koza, J.R., D. Andre. "Evolution of Iteration in Genetic Programming." *Evolutionary Programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming*. Cambridge: The MIT Press. In Press.

[9] Nordin, P., "A Compiling Genetic Programming System that Directly Manipulate the Machine Code." *Advances in Genetic Programming*. K. Kinnear, Jr. (ed.). Cambridge: The MIT Press, pp. 311-331.

Paper 5

Daida, J.M., T.F. Bersano-Begey, S.J. Ross, and J.F. Vesecky, Computer-Assisted Design of Image Classification Algorithms: Dynamic and Static Fitness Evaluations in a Scaffolded Genetic Programming Environment, 279-284, **Genetic Programming 1996: Proceedings of the First Annual Conference**, J.R. Koza, D.E. Goldberg, D.B. Fogel, and R.L. Riolo (eds.). Cambridge: The MIT Press (1996d).

To appear in the *Genetic Programming 1996: Proceedings of the First Annual Conference*, July 28–31, 1996, Stanford University, J.R. Koza, D.E. Goldberg, D.B. Fogel, R.L. Riolo (eds.) Cambridge: The MIT Press.

Computer-Assisted Design of Image Classification Algorithms: Dynamic and Static Fitness Evaluations in a Scaffolded Genetic Programming Environment

Jason M. Daida, Tommaso F. Bersano-Begey, Steven J. Ross, and John F. Vesecky
The University of Michigan
Artificial Intelligence Laboratory and Space Physics Research Laboratory
2455 Hayward Avenue, Ann Arbor, Michigan 48109-2143
daida@umich.edu, tombb@umich.edu, stevross@umich.edu, jfv@umich.edu

ABSTRACT

This paper discusses several issues in applying genetic programming to image classification problems in geoscience and remote sensing. In particular, this paper examines the role in using dynamic and static fitness evaluation functions. This paper also examines a few of the aspects in human-computer interactions that facilitate computer-assisted learning and problem solving (i.e., scaffolding) for our system. We describe a possible means for visualizing and summarizing a solution space without having to resort to an exhaustive search of individuals.

1. Introduction

1.1 Background

Classification (alternately, *pattern classification*, or *pattern identification*) denotes a process in which a decision rule is applied to categorize a set of image data (Haralick and Shapiro (1993)). This process often represents a key step in transforming image data into information. Francone et al. (1996) suggest that genetic programming (GP) can be used to evolve robust classification algorithms. However, there are certain trade-offs that are involved in using GP for classification tasks with image data, particularly with satellite image data used in the fields of geoscience and remote sensing. Of these, one key trade-off involves computational effort and human supervision.

In an "ideal" world, image classification tasks would require minimal, if any, human supervision and most of the programming would be done by a computer. Unfortunately, the computational effort associated with classifying satellite image data is nontrivial. A typical satellite image may consist of one to hundreds of megabytes, which often involves a fair amount of CPU time for processing. Common geoscience problems can also often involve the classification of many images. When one considers that GP involves the evaluation of many candidate solutions, each of which could require seconds to hours of computation per image, the prospect of using just one image for evaluation becomes daunting; the cost of using just one image as a fitness set becomes prohibitive. Of course, one could reduce the amount of computation involved by reducing the size of the fitness set from whole images to subimage samples.

Although the idea of using subimage samples may seem straightforward, in practice, it is not. One often encounters situations where there exists incomplete knowledge about the variations in the features to be classified. In such situations, what exactly constitutes the minimal set that completely describes the categories of interest is unknown. It becomes entirely possible that a subimage sample may in fact represent an ill-posed problem. Moreover, the available computational resources might further constrain the size of a subimage sample to such an extent that the only practical sample corresponds to an ill-posed problem. In either circumstance, a workaround involves human supervision.

In Daida et al. (1996), we introduced a GP system for the computer-assisted design of image classification algorithms. The system has been used to solve a difficult pattern recognition problem that has had no prior satisfactory solution. The system has been designed to classify megabyte-sized gray-scale images, even though the corresponding fitness set may consist of only a handful of manually classified pixels (test points). The system has featured a canonical GP with a dynamic evaluation function (i.e., a function that increases the number of the fitness cases that are evaluated during the course of a run).

In designing this system, we have presumed the following. First, there exists incomplete knowledge on the user's part about the classification problem at hand. Second, there exists a high probability of a user formulating an imperfect fitness set (at least initially). Third, that GP provides "logical" outcomes to a presented fitness set. Fourth, that learning on the user's part about the classification task at hand would result in the crafting of better fitness sets, which would in turn generate better solutions. In this sense, GP is used as a kind of agent that handles the task of programming while the user is left to concentrate on the task of program specification. In other words, GP "scaffolds" a user to learn about the problem's essentials by illuminating consequences of specifications made with incomplete knowledge, while hiding the potentially distracting details associated with programming.

Although this system has performed well, we left for later the task of qualifying the necessity of using GP in a scaffolded fashion. In particular, this paper shows just how ill-posed fit-

ness sets can be, if only because large fitness sets are not computationally tractable. We also left for later the task of qualifying the reasoning for using dynamic, as opposed to static, fitness evaluation functions. This paper subsequently addresses these issues.

1.2 Previous Work

We have observed that most work in GP has not required scaffolding, since fitness, as defined by an evaluation function, has usually been sufficient in determining program success. We note that although we have not yet found other works using GP as a scaffolding technology for image analysis, we have found a similar use of GP in the area of computer-produced art (e.g., Sims (1991)).

The idea for using a dynamic evaluation function—one in which either the fitness cases or the evaluation function itself changes during the course a run—is not new. Koza (1992) used such functions in many examples. However, the particular strategy on which we based our work stems from Goldberg's (1989) work in genetic algorithm classifiers and Holland's (1986) work in default hierarchies and learning. Of interest to us has been how their strategies applies to image analysis problems that involve using GP to extract information from image data. Although Tackett (1993) has used GP with this type of image analysis problem, he used a different evaluation strategy. To date, we have encountered few works that evaluate the effectiveness of Holland's and Goldberg's type of dynamic evaluation function with GP systems for image analysis. We do note, however, that Schoenauer (1996) has used this type of evaluation function to increase the numerical precision of computational models. (See also Siegel (1994) and Angeline and Pollack (1993).)

2. Highlighted Methods

This section details two notable distinctives about our system described in Daida et al. (1996) and briefly mentions how these distinctives relate to the experiment discussed in this paper.

2.1 Dynamic Fitness Evaluation

For our system, dynamic fitness specifically refers to evaluating a set of program benchmarks that changes in membership during the course of a GP run. The following paragraphs supplement the qualitative description given in Daida et al. (1996).

Let P_0 represent the set of initial test cases, i.e.,

$$P_0 = \{a_0, a_1, \ldots, a_M\},$$

where a_i represents a test case (an image test point in our case), and M is a positive integer.

Furthermore, let P' denote the set of additional test-case arrays, i.e.,

$$P' = \{\{b_{00}, b_{01}, \ldots, b_{0I(0)}\}, \{b_{10}, b_{11}, \ldots, b_{1I(1)}\}, \ldots, \{b_{K0}, b_{K1}, \ldots, b_{KI(K)}\}\},$$

where P' is a K-element array of arrays (which do not necessarily have a uniform length), b_{kl} is a test case, and $I(k)$ is (integer length - 1) of the kth test-case array.

Given an evaluation function f such that

$$f(c_{individual}, P, t, trigger): \\ c_{individual}(p(t, trigger)) \rightarrow raw\ fitness\ score,$$

where $c_{individual}$ denotes an individual from a population of programs, $P = p(t, trigger)$ denotes the set of test cases to be evaluated by $c_{individual}$ at a time t, let p be a mapping such that

$$p(t, trigger) = \begin{cases} P_0, & t = 0. \\ P_0 \cup P(1), & t_1 < t < t_2, \text{ where } t_1 \text{ represents the first time } trigger = TRUE \text{ and } t_2 \text{ represents the second time } trigger = TRUE. \\ P_0 \cup \bigcup_{k=1}^{K} P(k), & t_K < t < t_{K+1}, \text{ where } t_K \text{ represents the } K\text{th time } trigger = TRUE \text{ and } t_{K+1} \text{ represents the next time } trigger = TRUE. \\ \cup \bigcup_{k=1}^{K} P(k), & t_K < t \le t_{Max}, \text{ where } t_{Max} \text{ represents the total amount of allowable time steps.} \end{cases}$$

For our purposes, we let t denote generational time, e.g., $t = 0$ denotes generation zero. (Note: function p can also be expressed in terms of individual time, which would be relevant for some cases of adaptation.) The Boolean trigger in function p represents the conditional in which the test-case set is incremented with new test cases. Although there are several different ways to trigger an increment, for both this paper and in Daida et al. (1996), the trigger used was strictly performance based. In particular, an increment was triggered when enough hits were scored for a particular set P, i.e.,

$$trigger = \begin{cases} TRUE, & (Number\ of\ Hits) \ge \\ & (\text{Length}(P) - Margin). \\ FALSE, & \text{Otherwise.} \end{cases}$$

We denote $Margin$ as a parameter that a user defines, where $Margin$ is an nonnegative integer such that $Margin \le Length(P)$ for all sets P.

2.2 Scaffolding

Figure 1 shows a flowchart of how we have used GP in a system for the computer-assisted design of image classification algorithms. For this paper, we deliberately chose not to use the full configuration in Daida et al. (1996), but instead allowed GP to execute without human supervision for two series of runs (i.e., feedback loops indicated by the dashed gray lines were not used). (This scenario could happen in actuality: a user would execute a series of GP runs to get a probabilistic assessment of her hypothesis before she would amend any inputs to the GP kernel). This partial configuration was used for this paper to test scaffolding and fitness evaluation functions.

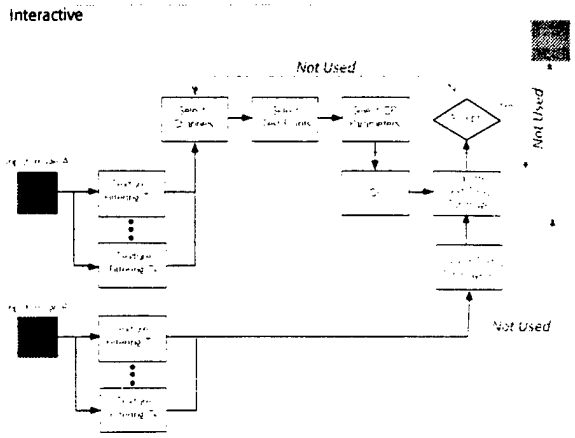


Figure 1. Described Scaffolded GP System.

3. Procedure

3.1 Domain Problem

The domain problem featured in this paper is identical to that described in Daida et al. (1995, 1996). The problem involves extraction of light-gray curvilinear and blob features from a light-gray background. The images are synthetic aperture radar (SAR) low-resolution data products that depict scenes of the Arctic Ocean and surrounding seas. The particular SAR of interest is mounted on the European Space Agency ERS-series of satellites. The curvilinear and blob-like features in these scenes of the ice cover correspond to pressure-ridges and rubble fields, respectively. This particular problem has been difficult to solve because of the low signal-to-noise ratio of the desired features to background. There has been no record of an algorithm that satisfactorily solves this problem for ERS SAR imagery prior to Daida et al. (1996).

3.2 Experiment Design

The dynamic case portion of this experiment consisted of executing 20 runs of system configuration shown in Figure 1 with the dynamic evaluation function described in Section 2.2. Each run used a different random number seed. The static case portion of this experiment consisted of executing 20 runs of GP with a static evaluation function. Identical random number seeds were used for both dynamic- and static-case runs. All other common parameters, terminals, functions, and an out-of-sample evaluation image remained constant. Table 1 summarizes some of those items.

Note that the following items have been presented in greater detail in Daida et al. (1996). Information in parentheses refers to where in that work a description for a particular item is given, namely: terminal set (Figure 21.4), function set (Tables 21.1–21.3), and image data (in particular, the April 23, 1992 data shown in Figure 21.3b). We further note that implementing the function set for closure under eight-bit image data does involve more than what can be conveyed in this paper. For example, addition can be implemented in at least two ways: saturation 255 or modulo 255. (Under saturation 255 arithmetic, the sum of any number greater than 255 is 255.) We included both implementations in the function set as P and +, for saturation 255 and modulo 255 addition, respectively.

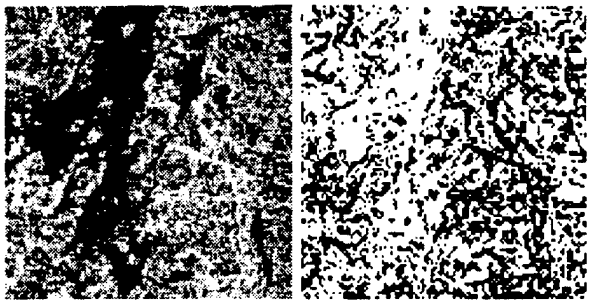
The following parameters were used for both dynamic- and static-case portions: maximum number of generations were equal to 50; size of population, 300; maximum depth of new individuals, 6; maximum depth of new subtrees for mutants, 4; maximum depth of individuals after crossover, 17; fitness proportionate reproduction fraction, 0.1; crossover at any point fraction, 0.2; crossover at function points fraction, 0.7; selection method, fitness-proportionate; generation method, ramped-half-and-half. Note that the population size and number of generations used were modest and chosen in part so that we could complete a run on an HP 715 within a few CPU hours.

The following parameters were used for the dynamic evaluation function: 15 initial test points, subsequent increments of nine test points, *Margin* of six hits. Note that both dynamic and static cases used the same set of test points. The most significant difference was that the dynamic evaluation function made the test points gradually available by small increments, while the static evaluation function made all 53 test points available as a single chunk.

Figure 2a shows the out-of-sample test image that was used. This 128×128 pixel image is a subset from the original non-geocoded April 23, 1992 ERS SAR data product that has been calibrated in gray scale. (Note: image attributes of geocoding and calibration are mostly for domain science purposes. Geocoding refers to image data that has been geometrically corrected according to a specified latitude-longitude grid. Calibration allows for direct comparison between images taken from different times and different satellites.) The subimage shown in Figure 2a has been contrast enhanced for publication. For comparison purposes, Figure 2b shows the classification results using the best known algorithm (from the switch filter, described in Daida et al. (1996)).

Table 1. Settings and Parameters

Terminal Set:	Consists of an array of size N of manually selected test points, which contains data from an image and its corresponding filtered versions (i.e., 5×5 Laplacian of a 3×3 Mean, 5×5 Laplacian, 3×3 Mean, 5×5 Mean); a random floating-point variable.
Function Set:	Arithmetic operators defined to operate on matrices and constants in any combination (+, −, ×, ÷, P, M), and the threshold operator If-Less-Than-or-Equal (≤).
Fitness Cases:	List of N manually selected control points from benchmark. Pixels that are part of pressure ridge features are given a gray-scale value of 255 and all other non-feature pixels take a value of zero. The number of current fitness cases depends on the type of evaluation function used.
Raw Fitness:	The number of hits.
Standardized Fitness:	(Number of fitness cases, N) - (Number of hits).
Hits:	The number of fitness cases for which an individual program's output is less than 9 away from the target gray-scale value of 255 or 0.
Success Predicate:	A run ends when the maximum number of generations is reached or when an S-expression scores the maximum number of hits.



(a) (b)

Figure 2. (a) Subimage from April 23, 1992. Image of sea ice near Beaufort Sea gyre. ERS-1 © ESA 1992. (b) Reference classification.

3.3 Experiment Rationale

The experiment design of Section 3.2 allowed us to:

- isolate the role of the evaluation function by completely removing human supervision from fitness evaluation. In particular, it allowed us to ask whether supplying test cases in toto or by increments would make for significant differences in classification output. We have hypothesized that a dynamic fitness function would likely yield better solutions than a static fitness function, given identical sets of fitness test cases.
- examine the case where no scaffolding is allowed. Although our use of scaffolding does indicate that a human is involved in some aspect of fitness evaluation, we have designed the system to limit that involvement mostly to actions taken between GP runs (i.e., evaluations of out-of-sample results, as well as changes made to fitness cases and GP parameters). Still, we have hypothesized that even this amount of restricted access is (for all practical purposes) necessary because the problem is likely to be ill-posed. For this case where no scaffolding is allowed, one can test for this hypothesis by showing that the out-of-sample results are not well correlated to fitness scores.

3.4 Implementation Notes

Most of the implementation did not change from that described in Daida et al. (1996). All of the runs were concurrently executed on a network of HP 715 UNIX workstations using Allegro Common LISP.

Each run takes about four hours to complete. Although the workarounds described in Daida et al. (1996) can and have allowed for the processing of 1024×1024 pixel images, we chose to use 128×128 pixel subimages for out-of-sample inspection. We have found that this size of subimage is sufficient to inspect all of this experiment's images on a monitor without having to sacrifice the detail necessary for a qualitative assessment. (At least this is true for our domain problem. Other problems may require larger subimage sizes than what we used.)

4. Results

Figure 3 shows the performance curves associated with the experiment. All of the curves are displayed as box plots, which summarize the statistical variability measured for each generation.

Figures 3a and 3b show the box plots associated with the average standardized fitness measured for each generation for static and dynamic evaluation functions, respectively. A perfect standardized fitness score is zero hits and the worst case score is 53 hits, which is indicated by dotted lines. Figures 3c and 3d show the box plots associated with the raw fitness score for the best individual of each generation for static and dynamic evaluation functions, respectively. A perfect score is 53 hits, which is also indicated by dotted lines.

Figures 4 shows an excerpt from the out-of-sample test image for static and dynamic evaluation functions. For each row, each image corresponds to an output from a best-performing individual from a single run. Only the first individual that the kernel identified as having the highest raw fitness score has been visualized. (A run can generate many individuals with the same score.) Black (gray value 255) represents regions that supposedly have pressure-ridge or rubble-field features. The images in each row are organized by trial so that the main difference between images in a column is the type of evaluation function (and not a random-number seed). Visualized are the results corresponding to the first ten random-number seeds.

5. Discussion

5.1 Interpreting Performance Curves

The shapes of the performance curves are the most different at the outset, from zero to about seven generations. According to Figure 4, the static evaluation function allowed for GP to create an individual that solves about 75% of the presented problem (approximately 38 hits in raw fitness score) in generation zero, which indicates that most of the solution has been generated by a pure random process. In contrast, the dynamic evaluation function allowed for the GP to solve at most 30% of the presented problem (15 hits, raw fitness score) also by a pure random process. The short rise of the average standardized fitness curve for the dynamic case suggests that the presented problem has increased in difficulty for several generations. Figure 4 suggests that one effect, then, of using dynamic evaluation was to shift the predominant means of obtaining a solution from a pure random process to one of crossover and reproduction.

The shapes of the performance curves are the most similar towards 50 generations. The particular fitness set used in this experiment did yield an individual with either a perfect or near-perfect score for all of the runs, whether static or dynamic.

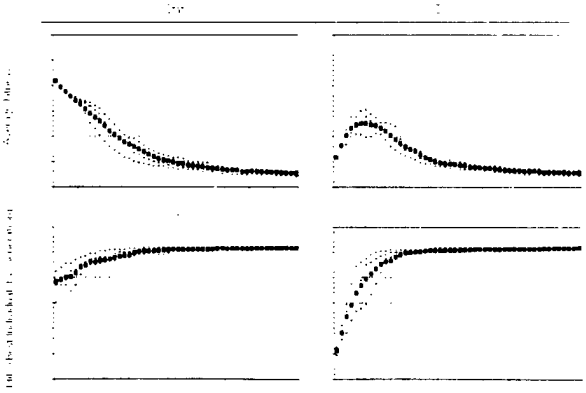


Figure 3. Performance curves.

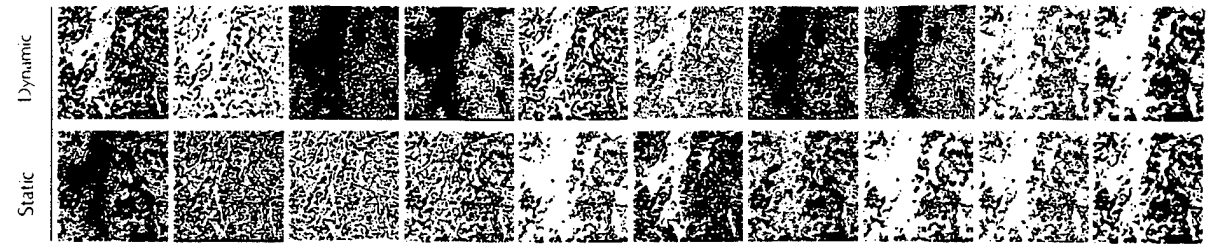


Figure 4. Excerpt of Out-of-Sample Results.

5.2 Classification Output and Scaffolding

The results presented in Figures 3 and 4 indicate that the out-of-sample classification results were not well correlated with high fitness scores (which is an expected finding for this experiment). At face value, the performance curves shown in Figure 3 would have suggested that all algorithms obtained from runs under either static or dynamic fitness evaluation functions correspond to reasonable classification solutions. However, when these algorithms classified imagery that contained data exclusive of that used in fitness calculations, the quality of classification output has varied greatly. Even perfectly scoring algorithms yielded questionable results that looked nothing like the solution shown in Figure 2b. These results did strongly suggest that the provided fitness set was ill-posed, which is what we expected for the case of no scaffolding, and which is why we have advocated the use of scaffolding.

The out-of-sample processing requirement for image analysis tasks is often many times larger than the number of in-sample test cases with which GP can feasibly process. Considering the out-of-sample requirement for just *one* ERS low-resolution (1024×1024 pixel) data product, the ratio of the number of out-of-sample cases to the number of in-sample cases for our fitness set is almost 20,000:1. (In an operational environment, a 2,000,000,000:1 ratio would not be unreasonable.) Even if one could accurately and precisely quantify every relevant aspect of a feature for processing (which is unlikely at the start of problem solving, *cf.* Congalton (1991)), the possibility of undersampling would still remain. For that reason, using GP as a means for scaffolding (which would include a human in an evaluation loop with GP) in image analysis tasks makes more sense than using GP without human supervision.

5.3 Likelihood of Success

Are there any predictors that might indicate whether a reasonable solution is likely? Fortunately, the answer may be yes. To assess whether a set of parameters, terminals, and function sets might yield a solution, a user would likely end up executing a large number of runs, if only because GP is a probabilistic method. Furthermore, for ill-posed cases, a single run can generate upwards of several thousand individuals that have scores that would merit human inspection. In any case, without a predictor, a user could be faced with a fairly laborious task of inspecting each candidate without necessarily being assured of whether any solution would be possible for that particular set of GP parameters or fitness test cases.

Figures 5a and 5b show the result of doing a matrix addition and normalization over the twenty images for dynamic and static cases, respectively. White represents low average values; and black, high values. What has surprised us was how close the resulting averaged images were to the solution shown in Figure 2b. Part of this surprise came from noting that none of the images in Figure 4 looked like the solution. Indeed, some of those images looked quite different. Another part of this surprise came from also noting that the number of images averaged was fairly modest, given a considerable number of individuals (e.g., over 5000 in the dynamic case) having similar fitness scores.

We have hypothesized that each image represents the outcome of one or more genotypic building blocks that have evolved over the course of a run. The expression of those building blocks would depend on what building blocks exist and what blocks become expressed, which depend upon the structures that link these blocks together. Because components in the function set were Cartesian operators, an average image would represent a projection in image space of those building blocks that commonly occur. An average image would subsequently provide a rough indication of allowable and disallowable solutions.

If this hypothesis were the case, thresholding an averaged image would then yield a likelihood map of classified image data for some probability (i.e., given M runs, the probability that all the components necessary for a solution are attained). Assuming that the values associated with the probability of assembling all necessary building blocks remains relatively constant, the most likely map of classified image data should be comparable between static and dynamic cases for some fixed threshold.

Figure 6 shows the result of thresholding the averaged images. Figure 6a shows a subimage of the original data shown in Figure 2a. Figure 6b shows the corresponding classified solution for reference purposes. Figure 6c shows the thresholded averaged image for the dynamic case, while Figure 6d shows the thresholded averaged image for the static case. The thresholded values corresponding to Figures 6c and 6d were kept identical.

Figure 6 indicates that the dynamic evaluation function has a

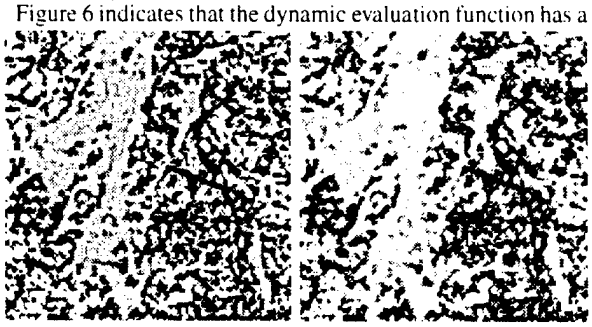


Figure 5. Averaged outputs. (a) Dynamic. (b) Static.

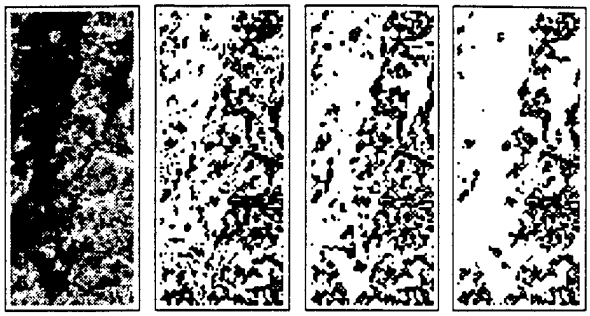


Figure 6. Probablistic outputs. (a) Data. (b) Reference solution. (c) Dynamic. (d) Static.

greater likelihood of obtaining a solution in comparison with the static evaluation function (i.e., Figure 6c is more similar to Figure 6b than is Figure 6d). What is missing in the static case is the building block(s) associated with classifying ridge and rubble features at the edge of multiyear ice. In other words, the probability of obtaining code that detects just the presence of bright curvilinear features at the boundaries between light gray and dark gray regions appears to be fairly low for the static case. Note that the required code would not be an edge detector. (If it were, all the edges between the light and dark gray regions would be incorrectly classified as ridge features.)

At the time of this experiment, we did not have the means to track building blocks for entire populations of algorithms and consequently did not have the means for testing this hypothesis. We have subsequently left this assignment for future work.

6. Conclusions

This paper has examined some of the issues involved in using GP in the computer-assisted design of image classification algorithms. Although it would be desirable to have the computer to do much, if not all of the work in deriving an appropriate classification algorithm, for practical reasons, we have indicated that a human should be involved in at least part of fitness evaluation. In particular, we have indicated that there is a trade-off in the number of fitness cases considered during evaluation versus the available computational resources. We have also pointed out that there may exist uncertainty about the quality of fitness cases used. In any case, we have indicated than an image classification problem could be ill-posed, which was certainly the case for this paper's experiment.

This paper has also has described the role of using either dynamic and static evaluation functions in deriving a solution. Results have suggested that use of a dynamic evaluation function would be more likely to evolve a classification solution than would a static evaluation function using the same set of fitness test cases. A novel means of visualizing and predicting the solution space for a given set of parameters, functions, and terminal sets was also described.

Acknowledgments

This research has been partially funded by grants from the Office of Naval Research, the Naval Research Laboratory (Stennis) and the Space Physics Research Laboratory (U-M). We appreciate the thoughtful critiques given to us by the reviewers of this paper. We gratefully acknowledge I. Kristo, P. Angeline, M. Bassin, S. Daida, J. Hommes, J. Koza, R. Onstott, R. Riolo, E. Soloway, and A. Wu for their invaluable comments and assistance.

Bibliography

Angeline, P.J., & J.B. Pollack, 1993. Competitive environment evolve better solutions for complex tasks. In S. Forrest (ed.), *Proceedings Fifth International Conference on Genetic Algorithms*. San Mateo: Morgan Kaufmann Publishers, Inc., pp. 264–270.

Congalton, R.G., 1991. A review of assessing the accuracy of classifications of remotely sensed data. *Remote Sensing of the Environment*, **37**:1, pp. 35-36.

Daida, J.M., J.D. Hommes, S.J. Ross, & J.F. Vesecky, 1995. Extracting curvilinear features from synthetic aperture radar images of arctic ice: Algorithm discovery using the genetic programming paradigm. *Proceedings 1995 International Geoscience and Remote Sensing Symposium: Quantitative Remote Sensing for Science and Applications*, IEEE Press., pp. 1415–1417.

Daida, J.M., J.D. Hommes, T.F. Bersano-Begey, S.J. Ross, & J.F. Vesecky, 1996. Algorithm discovery using the genetic programming paradigm: Extracting curvilinear features from synthetic aperture radar images of arctic ice. In P. Angeline and K. Kinnear, Jr. (eds.), *Advances in Genetic Programming II*. Cambridge: MIT Press.

Francone, F.D., P. Nordin, & B. Banzhaf, 1996. Benchmarking the real-world generalization capabilities of an advanced, compiling genetic programming system. *Proceedings Genetic Programming '96*. In press.

Goldberg, D.E., *1989. Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading.

Guzdial, M., 1995. Software-realized scaffolding to facilitate programming for science learning. *Interactive Learning Environments*, 4:1, pp. 1-44.

Haralick, R.M., & L.G. Shapiro, 1993. *Computer and Robot Vision: Volume II*. Reading: Addison-Wesley Publishing Company.

Holland, J.H., K.J. Holyoak, R.E. Nisbett, & P.R. Thagard, *1986. Induction: Processes of Inference, Learning, and Discovery*. Cambridge: MIT Press.

Koza, J.R., 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge: MIT Press.

Schoenauer, M., M. Sebag, F. Jouve, B. Lamy, & H. Maitournam, 1996. Identification of mechanical behavior by genetic programming: Part I rheological formation. In P. Angeline and K. Kinnear, Jr. (eds.), *Advances in Genetic Programming II*. Cambridge: MIT Press.

Siegel, E.V., 1994. Competitively evolving decision trees. In K. Kinnear, Jr. (ed.), *Advances in Genetic Programming*. Cambridge: MIT Press.

Sims, K., 1991. Artificial evolution for computer graphics. *Computer Graphics*, **24**:4, pp. 319-328, July 1991.

Tackett, W.A., 1993. Genetic programming for feature discovery and image discrimination. *Proceedings 2nd International Conference on AI Planning Systems*, pp. 303-309.